2013年11月号 总第2期

51CTO.com
技术博客 Blog

IT人专业博客
领先的IT技术博客

本期热点：

- 那些逝去的专业软件研发精神
- 防止人为误操作MySQL数据库技巧一例
- 我的微软最有价值专家(Microsoft MVP)之路
- 葵花宝典与学习之如何平稳的进入新技术领域

博客月刊

blog.51cto.com

目录

理解并演示：帧中继的逆向解析功能（frame-relay inverse-arp） 3

安全运维理念（半神半仙亦民工） 6

如何应对安全漏洞的修复 8

Hyper-V 3.0 部署 PART 10：无共享存储实时迁移 10

在 Microsoft System Center 中利用您的现有投资管理 VMware--Veeam MP v6.5 13

【虚拟化实战】VM 设计之一 vCPU 17

深入 Hadoop 节点部署的策略 20

Android 架构:认识简法设计与 EIT 软件造形 22

Zend Framework 1.12 中接受 ajax 传来的 json 数据 25

Tomcat 完美实现在线商城应用部署 26

C#设计模式(2)——简单工厂模式 30

C 语言进阶【文件】常用文件操作函数详解 33

防止人为误操作 MySQL 数据库技巧一例 35

论“软件测试实施” 36

葵花宝典与学习之如何平稳的进入新技术领域 38

那些逝去的专业软件研发精神 42

我的微软最有价值专家(Microsoft MVP)之路 44

理解并演示：帧中继的逆向解析功能（frame-relay inverse-arp）

作者：谌玺 来自：http://7658423.blog.51cto.com/7648423/1294309



微软 MCSE、思科 CCNA、CCIE 认证工程师，高校客作教授.培训过的微软 MCSE 系统工程师、思科 CCNA - CCIE 认证、企业网络整体安全、入侵检测与防预系统、防火墙与 VPN 技术、Winodws 服务器配置、Linux 服务器配置等课程。出版图书《企业网络整体安全--攻防技术内幕大剖析（附教学资源 》等。

理解并演示：帧中继的逆向解析功能（frame-relay inverse-arp）的配置对应的教学录像在：http://edu.51cto.com/lecturer/user_id-7648423.html9.9 发布，待管理员 24 小时审核后，便可观看，课程名称为《思科 CCNA 认证的帧中继专题分析（200-120) （9 课时）》

帧中继的逆解析功能（frame-relay inverse-arp）该功能用于提供一种动态的将二层 DLCI 号码映射到三层地址的一种方法。在本章的前面章节，对帧中继配置时，曾反复提到使用 frame-relay map ip 指令完成静态映射，申明使用本地的某个 DLCI 号映射到某个具体远程端的连接，有多少远程端，就要手工配置多少个 DLCI 号码静态映射远端 IP 地址的指令，在一些复杂的帧中继网络环境中会增加大量的管理开销，并且不能适应帧中继拓扑的变化，而帧中继逆向 ARP 的产生，将解决这些问题。事实上，帧中继的逆向 ARP 解析与局域网环境中的 ARP 地址解析协议非常相似，局域网中的 ARP 解析是知道目标 IP 地址后，去解析目标的 MAC 地址；而帧中继的 ARP 逆向解析是路由器将已知的二层地址（本地的 DLCI 号码）去映射远程端设备的三层 IP 地址。

第一步：如上图 8.73 所示，当帧中继路由器（DTE 设备）与帧中继交换机（DCE 设备）成功连接后，通过交换 LMI 消息，一台帧中继交换机可以宣告一条新的虚拟电路和该电路所对应的 DLCI 号码可用，但是它不能寻址到另一端，暂时无法完成与远程端的通信。

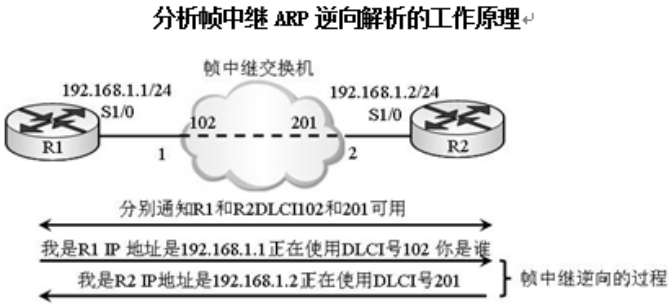


图 8.73 帧中继逆向 ARP 解析的工作原理

第二步：此时，如果帧中继的逆向 ARP 解析机制启动（在思科的网络设备上，它默认是启动的），路由器 R1 会发送一个关于帧中继的逆向 ARP 请求数据帧如下图 8.74 所示；“我是路由器 R1，我的 IP 地址是 192.168.1.1；我正在使用 DLCI 号码 102，请问你是谁？”



图 8.74 帧中继逆向 ARP 请求的数据帧

第三步：当路由器 R2 收到帧中继的逆向 ARP 请求后，它会对此作回应，回应数据帧如下图 8.75 所示，“我是 R2 三层的 IP 地址是 192.168.1.2。”然后路由器 R1 将该地址关联到本地的 DLCI 号（102）完成帧中继动态映射的过程。



图 8.75 帧中继逆向 ARP 回应的数据帧

演示：区别帧中继逆向ARP解析与静态映射的差异

演示：区别帧中继逆向 ARP 解析与静态映射的差异

演示目标：区别帧中继逆向 ARP 解析（动态映射）与静态映射的差异。

演示环境：如上图 8.73 所示的演示环境。

演示背景：首先在演示环境中，使用动态帧中继的逆向 ARP 解析来完成通信，然后禁用帧中继的逆向 ARP 功能，使用静态映射来完成通信，最终通过 show frame-relay map 指令来对比两种方式的差异。

演示步骤：

第一步：关于帧中继交换机的配置，前面已经有详细说明，这里就不再重复展示帧中继交换机的配置了，注意:思科设备上默认是已经启动了帧中继的逆向 ARP 解析（frame-relay inverse-arp）,所以路由器可以使用帧中继的逆向 ARP 功能来完成本地 DLCI 号码到远端 IP 地址的映射，这样，就不再需要使用 frame-relay map ip 指令完成静态映射，只需要在路由器 R1 和 R2 的帧中继接口上封装帧中继并配置 IP 地址即可，具体配置如下所示：

路由器 R1 的帧中继配置：

```
R1(config)#inte s1/0
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#encapsulation frame-relay
R1(config-if)#no shutdown
```

路由器 R2 的帧中继配置：

```
R2(config)#inte s1/0
R2(config-if)#ip address 192.168.1.2 255.255.255.0
R2(config-if)#encapsulation frame-relay
R2(config-if)#no shutdown
```

第二步：当完成上述配置后，可以让路由器 R1 ping 路由器 R2；如果一切正常，可看到路由器 R1 将成功 ping 通 R2，然后在路由器 R1 上执行 showframe-relay map 指令，来查看路由器 R1 的帧中继映射情况，如下图 8.76 所示，dynamic 指示动态映射；0x66 实际上是 DLCI 号 102 的十六进制表示，0x1860 是帧中继的帧封装中所表示的 DLCI 号，这个值将会因为各个位在帧中的置位方式不同而有所不同，事实上 0x66 和 0x1860,都是对 DLCI 号码的表示，只是显示方式不同而已，而 0x1860 显示方式是包括 DLCI、扩展地址 EA 及拥塞控制位的 16 位地址字段方式。



图 8.76 使用动态映射完成的状态

第三步：通过关闭设备上的帧中继逆向 ARP 解析功能（no frame-relay inverse-arp），然后使用静态映射的方式来完成帧中继的通信，具体配置如下所示：

路由器 R1 使用帧中继静态映射的配置：

```
R1(config)#interface s1/0
R1(config-if)#no frame-relayinverse-arp
R1(config-if)#frame-relay map ip192.168.1.2 102 broadcast
R1(config-if)#exit
```

路由器 R1 使用帧中继静态映射的配置：

```
R2(config)#interface s1/0
R2(config-if)#no frame-relay inverse-arp
R2(config-if)#frame-relay map ip 192.168.1.1 201 broadcast
R2(config-if)#exit
```

第四步：当完成上述配置后，可以再次让路由器 R1 ping R2,如果一切配置正常，此时，路由器 R1 应该能成功的与 R2 进行通信，然后在路由器 R1 上执行 showframe-relay map 指令，来查看路由器 R1 的帧中继映射情况，如下图 8.77 所示，static 指静态映射。

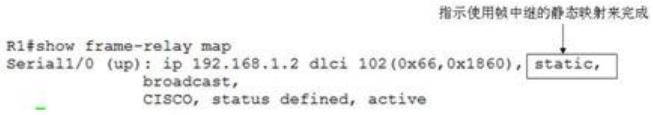


图 8.77 使用静态映射完成的状态

注意：在使用帧中继的逆向 ARP 解析时，如果远端路由器不支持逆向 ARP 解析，那么您只能使用帧中继的静态映射来完成配置，逆向 ARP 解析在思科的设备上默认是启动，但是当您已配置帧中继的静态映射，那么逆向 ARP 解析将自动关闭！

安全运维理念（半神半仙亦民工）

作者：cs312779641 来自：http://chenhao6.blog.51cto.com/6228054/1312801

前言：前段时间买了李洋博士的（防线）看了好久，受益匪浅，企业安全重则之重.

1. 知己知彼，百战不殆

既要了解敌人，也要了解自己，百战都不会有危险，不了解敌人而只了解自己，胜败各一半，又不了解敌人，也不了解自己，那么每战都有危险。

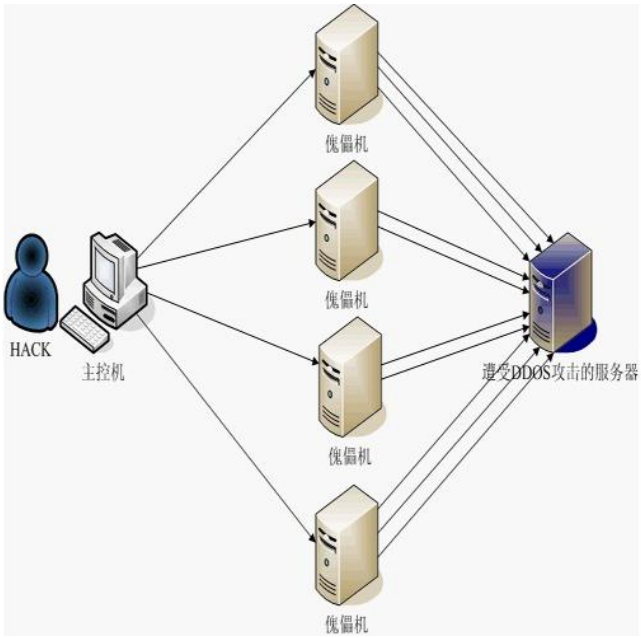
风险太大，安全运维需要有知己知彼的睿智，才能够在安全域威胁之争中取得先机，也能使企业在安全运维工作中立于不败之地。

对于企业运维来讲，他的对手只有一个 ----黑客。所以我们要了解黑客入侵所采用的的常用攻击手段和基本原理，这样我们才好防御。

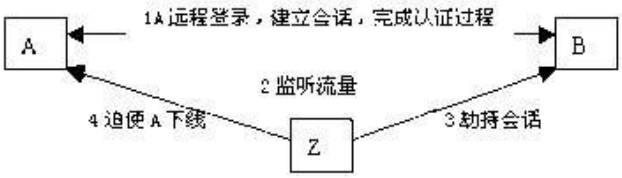
了解敌人：

1.dos 攻击 利用软件实现的权限，利用协议的漏洞，资源消耗

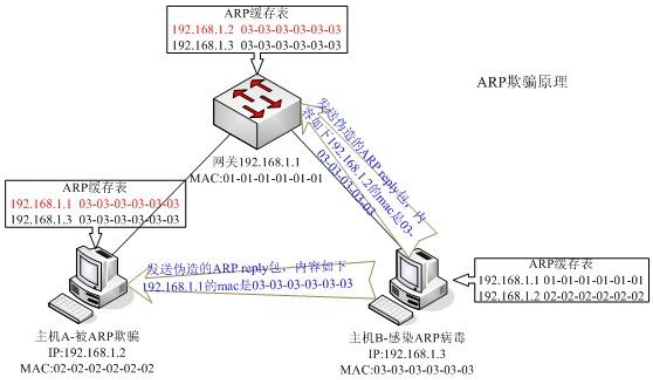
2.DDOS 攻击



3.IP 欺骗



4.ARP 欺骗



了解自己：

首先已经了解了敌人，那么我们就会做出相应的措施。

物理层防护：物理隔离

系统层防护：选用安全操作系统，操作系统密码复杂化，数据库安全技术

网络层防护：防火墙

网关级防护：UTM

2. 凡事预则立，不预则废

要想成就任何一件事，必须要有明确的目标，认真的准备和周密的安排，没有准备的盲目行动，只能是虽忙忙碌碌却也一事无成，有了精心的准备，艰苦的努力，不懈的奋斗，扎实的时间和巨大的付出，才能达到成功的彼岸。

企业安全建设和运维也是如此，需要从软硬件选型、信息安全建设的范围和内容等方面进行整体的规

划
数据（信息）安全
运行（系统）安全
物理（实体）安全
这三个方面考虑 规划部署！

3. 战略人藐视敌人，战术上重视敌人

在安全运维工作中，可以以之为指导思想，要把困难想在前面，做好周密的部署，要不畏艰难，从战略上藐视困难，耳聪战术上重视困难，做好相关的准备工作。

四步完成企业系统安全防护
1.linux 文件系统访问安全
2.linux 进程安全
3.linux 用户管理安全
4.linux 日志管理安全

4. 师夷长技以制夷

无论是管理员还是信息安全人员都需要站在黑客的角度来思考，从各个技术细节来对系统和网络可能遭受的攻击和外部威胁进行安全监控，这样才能在实际的运维工作处于有利地位，保证企业信息安全。

常用的性能检测工具：
top 处理器活动情况
uptime 系统平均负载
free 内存使用情况
pmap 处理器内存使用情况
ulimit 系统限制
mpstat 多处理器使用情况
vmstat 系统情况
dmesg 硬件情况
traffic-vis 网络监控
cacti 软件 可以收集 cpu 利用率，磁盘空间，web 上流量数据等所有信息

5. 工欲善其事，必先利其器

在企业 linux 安全运维中，其实有很多捷径，这包括掌握一些核心的理念，框架和方法，以及充分的利用现有的开源工具，命令，来灵活方便，高效的进行运维工作，这样提高工作效率。

1.定制内核
2.安全备份工具
backupPC
taper
xtar
3.系统管理工具：sudo
4.网络安全界的瑞士军刀：netcat
5.路由追踪工具：tracerouter
6.操作系统识别工具：xprobe
7.系统弱点发现工具：SATAN

结语：目前有很多东西没用过，正在努力学习提高自身技术。但是这本书对我帮助还是很大的，不过运维就是半神半仙亦民工。不过除了技术，还要有良好的沟通技巧，良好的人际关系，明确你的发展地位，遇领导要服软，服从领导安排，不能越级报告都是很重要的！

如何应对安全漏洞的修复

作者：xjsunjie

来自：http://xjsunjie.blog.51cto.com/999372/1311026

在大多数企业和公司，并没有专门的信息安全部和安全工程师，所以安全漏洞的修复责任就落到了运维部门的工程师头上，那么当你拿到了一份安全评估报告后，该如何应对安全漏洞的修复呢？

首先明确风险的类型和确定等级。基本上来说，风险可以分为网络服务、系统、应用程序、中间件、数据库类的；从等级上来说，还可分为高、中、低三大类。做好这一步工作很重要，方便你理清头绪，有的放矢，不会眉毛胡子一把抓。着重优先解决高风险



漏洞，根据安全影响的层面和客户要求适度处理中风险漏洞，低风险的漏洞可以适当忽略。

其次，进行汇报、时间和人力协调，做好实施方案。因为进行漏洞的修补和版本的相关升级，都会影响到生产上的应用和对内、对外的服务，因此必须跟上级领导汇报情况和可能的风险，得到领导的批准和支持。然后制定详细方案下发各相关人员，安排合适的时间窗口，并通知各部门做好协调和配合。

分类名	高风险	中风险	低风险	总计
Apache	2	48	46	96
RPC	1	0	9	10
BIND	0	2	5	7
Windows SMB	5	7	17	29
Snmpd	0	3	10	13
MS SQL Server	5	3	9	17
SSH	0	3	1	4
Samba	0	2	1	3
VNC	0	0	1	1
IIS	1	2	3	6
Oracle	5	17	11	33
MySQL	4	31	91	126

51CTO.com 技术博客 Blog

最后准备预案和及时反馈。升级和打补丁过程中，也会经常出现一些小问题和意外，重要的是要准备好预案，事先对可能会出现的问题进行预估和做好解决方案，避免出差错。另外在整个过程中，及时做好信息反馈很重要，有利于各方面的信息畅通和动作配合。

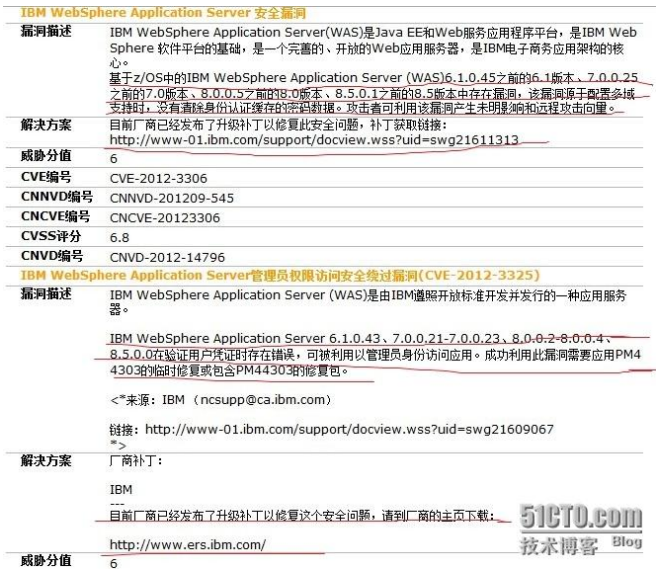
不管是安全漏洞的修复还是版本的升级，事先做好备份和应急预案，都是很重要的。牢记安全第一，防患于未然，这不仅是专业化的体现更是一种职业操守。

应用程序补丁安全升级方案示例：

前期分析



一般扫描和分析工具会列出具体的漏洞名称和风险级，你展开每个漏洞一般也会带有具体漏洞的信息



和解决方案，下面我们再看一个图，是绿盟给出的漏洞详细信息图。

注意图中划红线的部分，漏洞描述中分析了哪些版本会存在此问题并在下面的解决方案中给出了补丁下载信息。一般来说，你在一个应用中间件中发现了多个漏洞，并不需要一个一个的去修补，可以考虑合适的版本升级，比如以前是 7.0 版，现在可升级为 7.0 版本里的最高级。而如果是跨版本升级，则需要经过周密测试，一般在实际环境中不会贸然跨版本升级，这样做的话影响面大、风险也大。升级到一定版本后，针对其他的漏洞则可以单个的去修补。因此，经协商后，方案敲定为升级到 7.0 版本里的最高级，其他个别漏洞进行单一修补。

一、应用备份

备份前，检查确认是否需要扩充相关文件系统（主要保证有足够的空间），以便保证成功备份和软件成功安装。

将现有环境进行备份，主要是软件安装目录及相关配置文件

```
tar -cvf /backup/was.20130924.tar /usr/was/WebSphere/AppServer
tar -cvf /backup/http.20130924.tar /usr/was/HttpServer
```

二、软件和补丁安装

注意 WAS 软件和补丁安装使用 wasuser 用户。IHS 需要用 root ；

将补丁或新版本上传到相应的机器上，并设置正确的属主和权限。

WS-UPDI-AixPPC64.tar.gz

WS-IHS-AixPPC64-FP0000045.pak

WS-PLG-AixPPC64-FP0000045.pak

WS-WAS-AixPPC64-FP0000045.pak

设置正确的属主和权限

chown wasuser:wasgroup WS-UPDI-AixPPC64.tar.gz

chmod 644 WS-UPDI-AixPPC64.tar.gz

三、停止 WAS 和 HTTP 服务，应用在此期间将不能访问和使用。

停止 WAS

```
cd /usr/was/WebSphere/AppServer/profiles/Server1/bin
./stopServer.sh xxx -user wasadmin -password *****
./stopNode.sh -user wasadmin -password ******
su - wasuser
cd /usr/was/WebSphere/AppServer/profiles/Dmgr01/bin
./stopManager.sh -user wasadmin -password *****
```

停止 HTTPServer

```
/usr/was/HttpServer/adminctl stop
/usr/was/HttpServer/apatchectl stop
```

四、软件安装

在相应的机器上安装补丁，升级或打补丁完毕，在相应机器上启动 WAS 服务和 HTTP 服务。

在相应主机上安装 WAS UpdateInstaller

将补丁文件拷贝到

```
/usr/was/WebSphere/AppServer
gunzip WS-UPDI-AixPPC64.tar.gz
tar -xvf WS-UPDI-AixPPC64.tar
在主机上安装WAS补丁(需要图形界面进行安装)
cd /usr/was/WebSphere/UpdateInstaller
./update.sh进行安装，选择目录/usr/IBM/WebSphere，安装相应补丁即可。
在主机上安装HTTP补丁(需要图形界面进行安装)
cd /usr/was/WebSphere/UpdateInstaller
./update.sh进行安装，选择目录/usr/was/HTTPServer，安装相应补丁即可。
在主机上安装PLUGIN补丁(需要图形界面进行安装)
cd /usr/was/WebSphere/UpdateInstaller
./update.sh进行安装，选择目录/usr/was/HTTPServer/Plugin，安装补丁即可。
```

启动新版本服务

```
/usr/was/HTTPServer/bin/apachectl start
/usr/was/HTTPServer/bin/adminectl start
/usr/ was/WebSphere/AppServer/profiles/Dmgr01/bin/startDmgr.sh
/usr/ was/WebSphere/AppServer/profiles/AppServ01/bin/startNode.sh
/usr/ was/WebSphere/AppServer/profiles/AppServ01/bin/startServer.sh
clserver1
```

五、验证 WAS 服务正常

1、通过业务系统进行业务查询，验证 WAS 能够正常提供服务。

2、测试各应用和接口是否正常。

3、系统应用检查

测试功能和整体是否正常。

六、回退方案

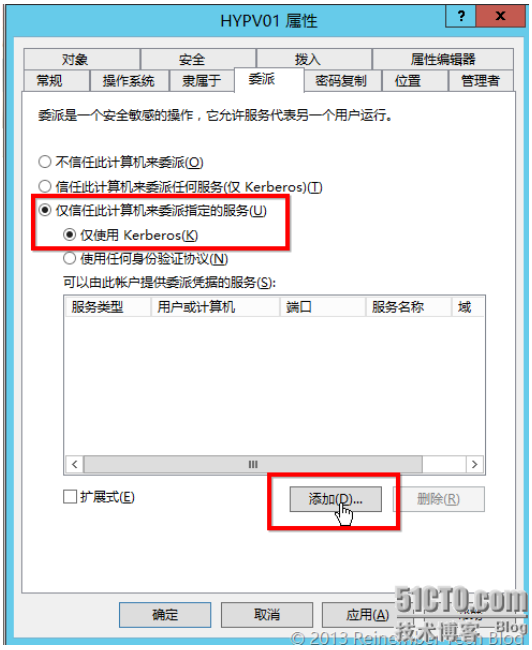
如果在 WAS 升级之后不能正常对外提供服务，需要停止新版本的 WAS 和 HTTP 服务，通过升级之前的备份来进行恢复到原来版本级别。

Hyper-V 3.0 部署 PART 10：无共享存储实时迁移

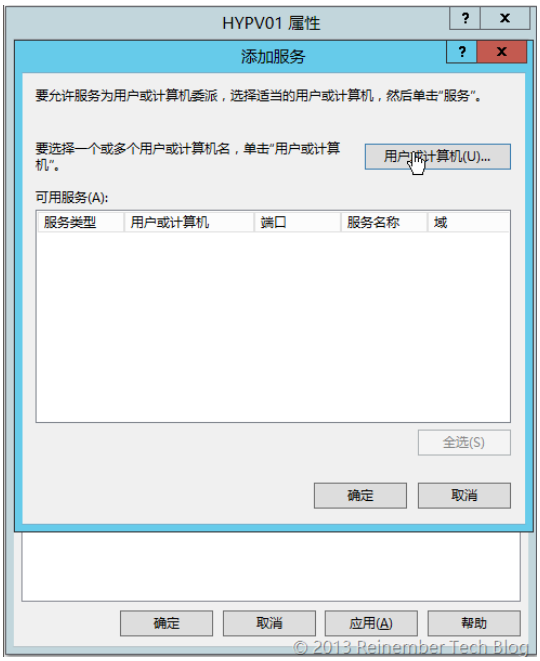
作者：reinxu 来自：http://reinember.blog.51cto.com/2919431/1307944

花费了很长的时间，总算把复制功能介绍了一下，最近实在是忙！今天有时间，我们再一起看一下 Hyper-V3.0 中，我们是如何实现不需要共享存储，直接进行实时迁移的。

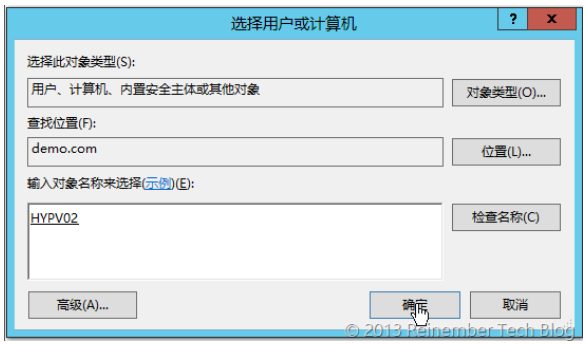
首先我们在 AD 用户和计算机中找到我们的 Hyper-V 主机，右键属性并选择委派，然后选择“仅信任此计算机来委派指定的服务”，然后下方选择仅使用 Kerberos，并单击添加。



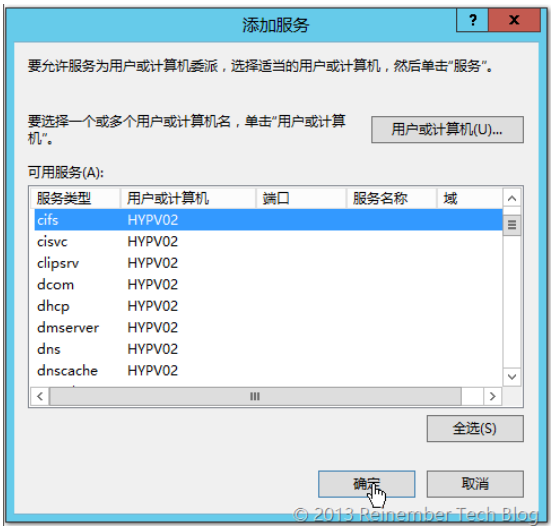
在添加服务对话框中，我们单击用户或计算机。



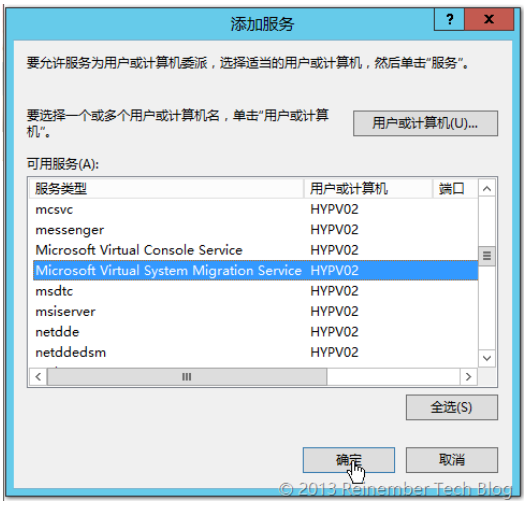
选择计算机，并把我们所有其他的 Hyper-V 主机全部键入，由于我的环境就 2 台 Hyper-V 主机，所以只用填写另外一台即可。



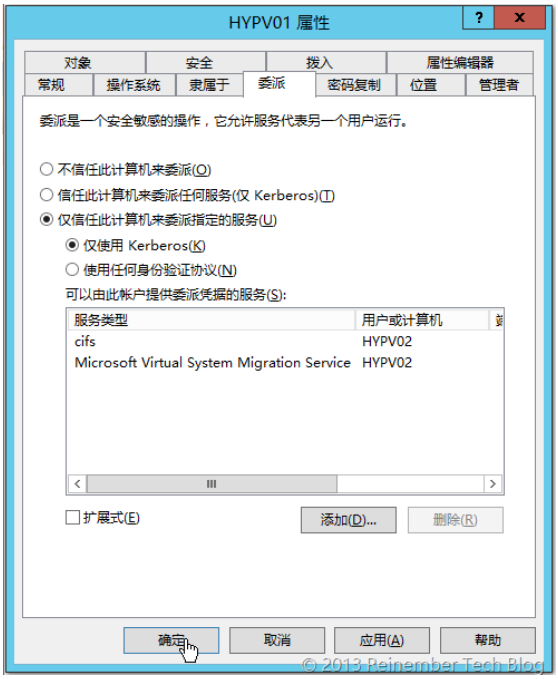
然后在可用服务列表中选择 cifs。



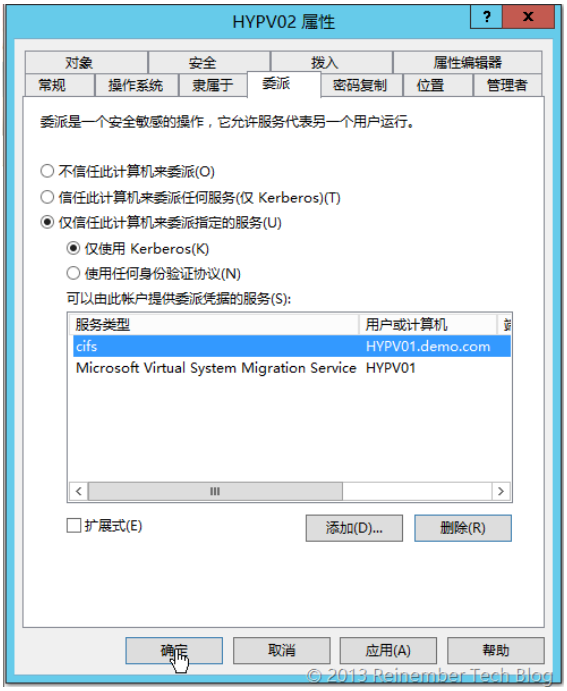
不要着急确定，接着往下找到 MVSMS，也就是微软虚拟系统迁移服务，英文名 MicrosoftVirtualSystemMigrationService。



回到属性界面，确保委派选项卡中的内容都是正确的，然后单击确定，保存这些配置。



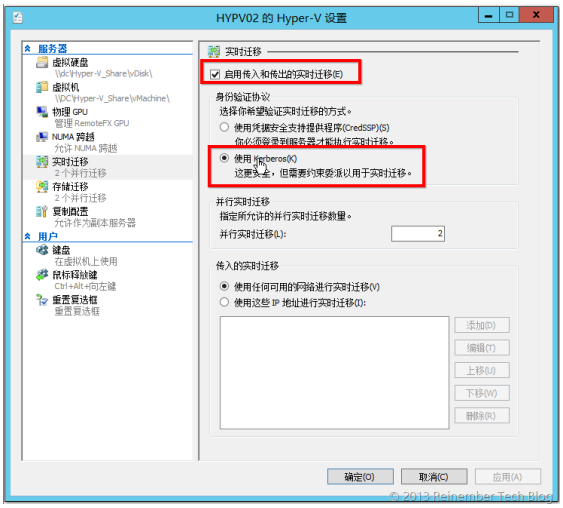
使用同样的方法，在其他的 Hyper-V 主机上完成相同工作。



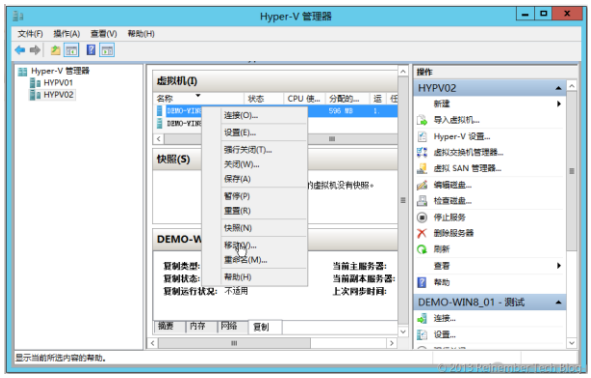
OK，回到我们的 Hyper-V 管理器，选择 Hyper-V 主机，并在右侧单机 Hyper-V 设置。



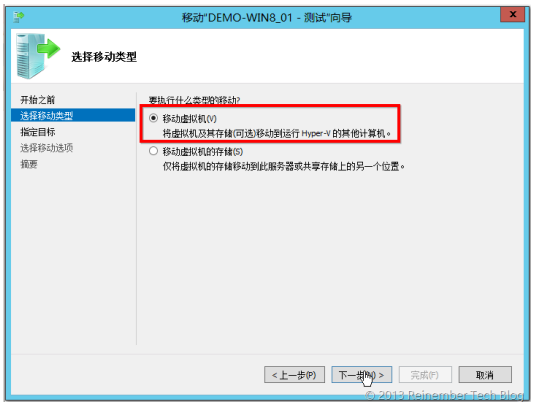
打开设置对话框后，选择实时迁移，然后选择“启用传入和传出的实时迁移”，然后身份验证协议选择使用 Kerberos。



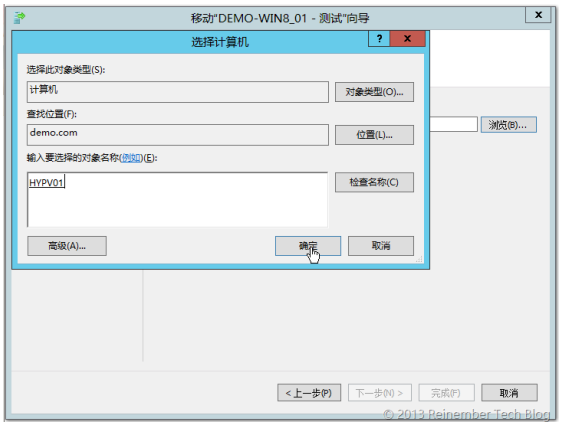
到这里我们就完成配置了，接下来我们来试下没有共享存储的迁移。我们在 Hyper-V 主机中随便找一台看着顺眼或者不顺眼的虚拟机，右键选择移动。



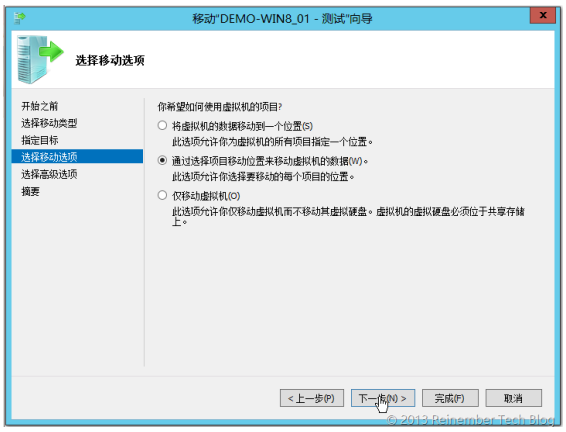
此时就会打开向导，我们选择移动虚拟机。



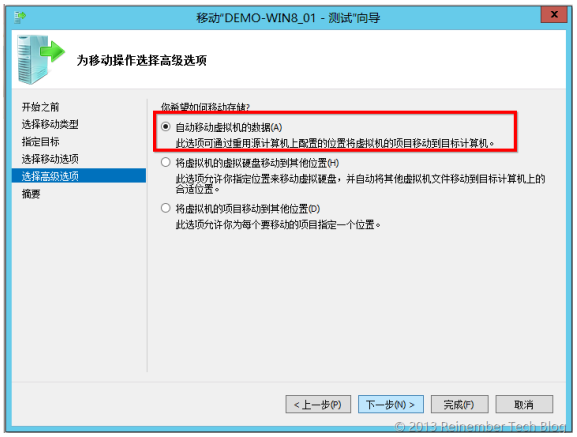
然后选择目标 Hyper-V 主机。



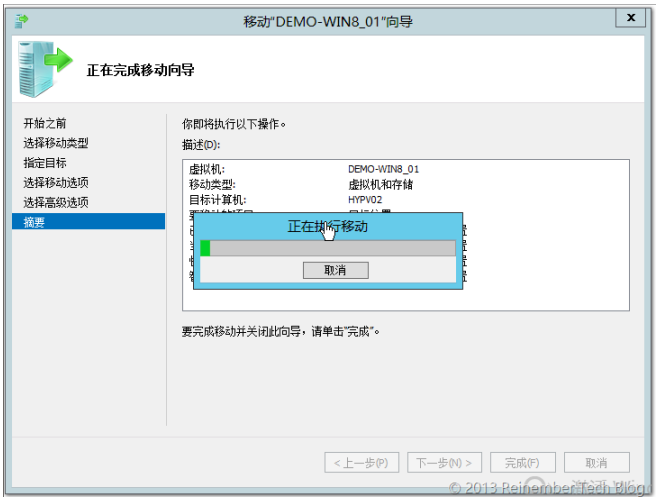
在“选择移动选项”中选择“通过选择项目移动位置来移动虚拟机的数据”，这样可能让我们选择存储而不是移动整个虚拟机。



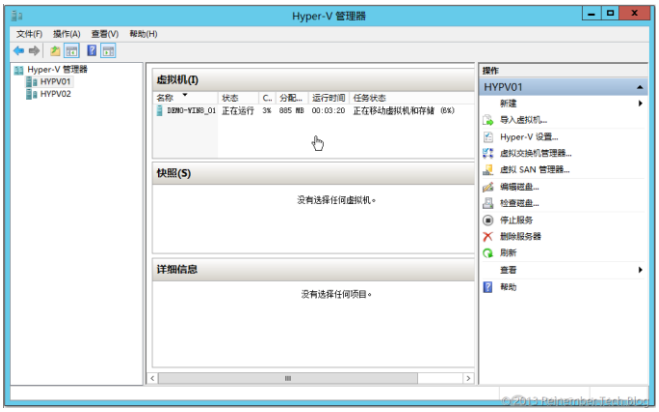
在选择高级选项，我们这里可以选择自动移动虚拟机的数据、也可以选择将虚拟机的虚拟硬盘移动到其他位置，如果仅仅是移动存储的话，那无疑就选择第二项。



开始移动。



我们在 Hyper-V 管理器中可以实时的看到移动的进度和状态。



无共享存储的实时迁移在某些情况下是非常有价值的，比如我们需要在两个群集间迁移时，可以直接把虚拟机的虚拟磁盘从群集 A 的存储迁移到群集 B 的存储，并且保证期间的迁移是不停机、不中断业务的。虽然迁移的时间会较长一些，但如果我们底层做了存储卸载功能，可以把这种迁移也提升到秒级，从而提供更多灵活的虚拟化平台管理手段。今天的内容就到这里，如果大家有什么想法、问题，欢迎随时回复文章，我们一起交流学习！

查看相关专题：
Hyper-V 3.0 功能部署
http://blog.51cto.com/zt/547

在 Microsoft System Center 中利用您的现有投资管理 VMware--Veeam MP v6.5

作者：linuxos 来自：http://winos.blog.51cto.com/937089/1310569

前段时间介绍了 Veeam Management Pack (MP) v6.0 产品，昨天发布了新版本 VeeamManagement Pack (MP) v6.5，将其新功能介绍如下：

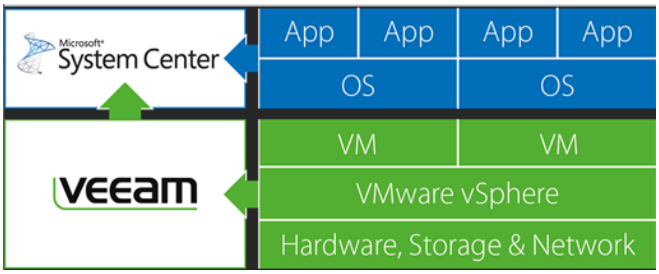
Veeam Management Pack (MP) v6.5 可利用您的 Microsoft System Center 投资将 VMware vSphere 整合到卓越的管理，其中包括适用于 Veeam Backup & Replication 的监控、报告和容量规划功能

全面查看 VMware 和虚拟机备份内容
Veeam Management Pack 提供一些主要优势，允许您：
n 在 SystemCenter 中充分利用您的现有投资
n 使用您的现有 IT 框架以包括 VMware
n 主动管理关键 IT 资源
l 容量规划
l 合理大小
l 优化
n 缩短平均解决时间(MTTR)
n 降低培训成本

Veeam MP 与 VMware 和 MicrosoftSystem Center 完全集成，能够为所有 VMware 组件（包括 vCenter Server、集群、主机、虚拟机、储存器和硬件）启用所有 Microsoft System Center 功能，其中包括报警、图表、仪表板、报告、审计、通知、响应和自动化等。Veeam MP 和 Microsoft System Center 共同提供一个一体化解决方案，以帮助您解决虚拟、物理、备份和应用挑战。

全新！使用 6.5 版，Veeam MP 可发现 Veeam Backup & Replication 环境以便直接在 OperationsManager 控制台中提供适用于备份基础架构的监控、报告和容量规划功能。

利用物理和虚拟基础架构和应用程序的统一视图，您可以快速找到并解决问题！将 Veeam MP 与 Microsoft System Center 搭配使用，可以通过使用单一的虚拟管理平台视图提供真正的卓越的可见性。没有什么比在 Microsoft System Center 中获得虚拟化可见性更好了！



VeeamMP 和 Microsoft System Center 可提供真正的卓越的可见性

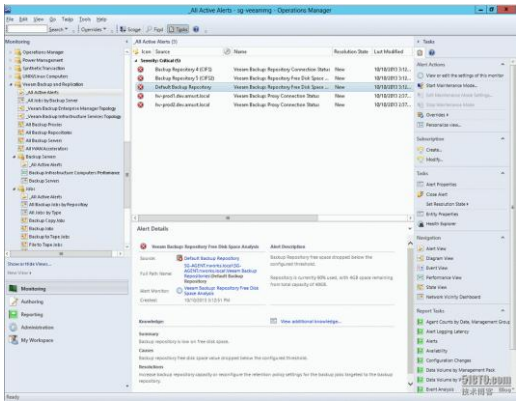
v6.5 中的新功能

Veeam Management Pack (MP) v6.5 将深入的 VMware 可见性扩展到 VeeamBackup & Replication 环境。Veeam Management Pack for VMware 的最新版本增添了三项重要的新功能：
n 适用于 VeeamBackup & Replication 的监控和报告
n 增强的容错能力
n 配置跟踪和警报关联

适用于 Veeam Backup & Replication 的监控和报告全新
如果使用 Veeam Backup & Replication，您将能够监控整个备份基础架构，包括代理服务器、存储库服务器、广域网加速器以及在备份服务器上运行的备份工作等。

Veeam MP 可发现 Veeam 备份环境，并利用针对每个工作的警报提供在备份服务器上运行的单个备份

工作的性能指标。此外，还可以监控备份工作，以便在工作出现故障后及其过程中发出警报（可选）。



VeeamMP 监控 Veeam Backup & Replication 使用的所有备份基础架构

Veeam MP 提供了先进的分析报告，其中包括：

n 受保护和未受保护的虚拟机

n 虚拟机验证

n 还原操作员活动

n 备份复制工作

n 适用于备份知识库的容量规划

增强的容错能力全新

Veeam MP 可确保持续提供用于自动重新定位数据收集的监控数据。如果 vCenter 出现故障，Veeam MP 可自动进行故障切换和重新配置，以便直接从 vSphere 主机收集数据。并且，在还原 vCenter 时，Veeam MP 可以在不中断的情况下返回到默认的数据收集方法。

配置跟踪和警报关联全新





此份新报告为您显示了所有 vSphere 对象的配置属性的详细跟踪，这样您就会知道在 vSphere 环境中发生了什么。此功能非常重要，因为如果在使用资源或规划容量时出现问题，您将可以利用事件的相关性帮助查明确切的原因。

Veeam MP 核心功能

以下这些功能是 Veeam MP 增添的最核心功能，其中包括：

n 用于深入分析信息的拓扑视图

nSystem Center 2012 仪表板

n 内置警报知识库

n 存储热图报告

n 容量规划

存储、网络和计算拓扑视图

分布式存储设备、网络和计算机环境的独立拓扑视图，能够让您快速浏览关键应用程序、服务以及支持的虚拟和物理基础架构之间的关系和依赖关系。您可以从拓扑视图直接链接至实时仪表板，并监控关键指标（如延迟、配置和使用情况）。

SystemCenter 2012 仪表板

仪表板视图显示了可以充分利用 System Center 2012 Operations Manager 小组件的关键 vSphere 系统的实时性能视图，例如每个集群的

“占用 CPU 的前十台主机”和每台数据存储设备的“占用磁盘输入/输出的前十台虚拟机”。

警报

综合性 VMware 知识库，可以将它用作完善的综合性 VMware 知识库，能让一线监控人员充分利用并缩短解决问题的时间。警报可显示 VMware 知识库文章中的信息，包括问题总结、原因和建议的解决方案。

报告

数据存储设备空间和数据存储设备性能使用情况的存储监控热图可提供即时的可视化表示，利用颜色和图形来显示基础架构的增长速度有多快。您可以清楚地看到最大的虚拟机、增长速度最快的虚拟机以及需要深入分析以查找的原因。数据存储设备热图以红色表示增长速度最快的虚拟机。

容量规划

Veeam MP 为 VMware 提供了四种可以直接整合到 Microsoft System Center 的容量规划报告：

l 主机故障建模

主机故障建模可显示如果在集群中主机发生故障 CPU 和内存将会发生什么

l 数据存储设备的性能预测

lvSphere 集群的性能预测

l 虚拟机容量规划

支持功能

现成的集成

Veeam MP 可以启用 Microsoft SystemCenter 的所有功能：为所有的 VMware 组件（包括 vCenter Server、集群、主机、虚拟机、存储器和硬件）启用报警、图表、仪表板、报告、审计、通知、响应和自动化。内置的阈值、图表和报告用于提供现成的含有数百项指标和事件的详细运行状况模型，其中包括内存压力、磁盘压力和磁盘 IOPS 等 Veeam MP 独有的先进指标。

监控

性能分析和相关性监控—这些监控类型可以接受输入多个性能指标数据，并从其他 vSphere 对象收集相关数据以帮助您更快解决问题。以外，它还提供了警报说明中的相应动态相关数据。

监控功能— Veeam MP 提供了多个性能指标（包括集群、资源池、主机和虚拟机的 CPU、内存、磁盘和网络 I/O）和事件（包括主机、虚拟机、资源池和集群状态、虚拟机迁移和部署、许可、存储和网络可用性）。

全天候提供关键的监控数据—通过利用具有自动化高可用性和负载平衡功能的分布式架构，连续提供有关 vCenter Server、vSphere 主机和虚拟机状态的丰富和详细的信息。

报告

合理大小报告—这些报告可用来指导为过大的虚拟机重新分配资源，以帮助您运行更高效的数据中心。

Veeam Extended Generic Report Library — Veeam 可利用 erweiterten generischen Report-Library von Veeam (GRL). 提供增强的 Microsoft System Center 报告。现在，您可以分析基础架构对象的运行状况及性能，而不必局限于物理或虚拟环境，或特定管理包。Veeam Extended GRL 可让您通过自定义新的参数和规则创建详细的报告。您可以对其进行扩展，因为目前的 Microsoft System Center Generic Report Library 不包括此功能。更完美的是，它是免费的。

安全和无代理数据收集

Veeam MP 利用最佳实践的应用程序接口方法从虚拟基础架构收集相关的事件、拓扑结构和性能信息。无需在 vSphere 主机或虚拟机中使用任何代理软件，便可削减在 hypervisor 中运行代理的开销、降低风险并缩短有限的使用寿命。此外，还可以对数据进行加密，并通过 HTTPS 进行安全传输。

可扩展部署和集中化管理

可以扩展多个分布式收集器以监控最大的 VMware 环境。并且，集中化 Web 控制台还简化了配置、许可证管理和升级过程。

支持的平台

Microsoft 平台

System Center 2012 R2 Operations Manager - 全新

System Center 2007 R2 Operations Manager 及更高版本

Windows Server 2012 R2 - 全新

Windows Server 2008 及更高版本

VMware 平台

vCenter Server 4.x 及更高版本 (5.5 版 - 全新)

ESX 4.x 及更高版本 (5.5 版 - 全新)

【虚拟化实战】VM 设计之一 vCPU

作者：frankfan7 来自：http://frankfan.blog.51cto.com/6402282/1316663



云计算/虚拟化专家 范军
全球第 93 位 VMware 认证设家（截至 2013 年该认证全球仅百余名）。15 年大型复杂 IT 环境的从业经验。参与过的项目覆 VMware ,Linux ,Citrix ,Google Apps ,Salesforce 等多项技术。曾受邀在 2013 年 VMware 合作伙伴大会中发表关于虚拟化设计和高级认证的演讲。

虚拟机需要多少个 vCPU 呢？是不是个数越多性能越好呢？这方面存在着很多误区。给 VM 配置 CPU 资源的时候，要精打细算才能最大可能的利用已有资源，来满足商业应用的需要。有的情况下为某个 VM 设置过多 vCPU 数目，反而会造成该应能的性能下降。也造成整个系统的资源浪费。

本文从概念到实战来阐述如何做好虚拟机上 CPU 资源规划。

一概念：

首先我们明确一些概念。

Socket

A CPU socket or a CPUpackage refers to a physical unit of CPU which is plugged into a system board.For example, a 4-way system or a 4-socket system can contain up to four CPUpackages

Core

A CPU socket or a CPUpackage refers to a physical unit of CPU which is plugged into a system board.For example, a 4-way system or a 4-socket system can contain up to four CPUpackages

Logical processor:

Hyperthreadingtechnology allows a single physical processor core to behave like two logical processors.The processor can run two independent applications at the same time. Aprocessor core may have multiple logical processors that share computeresources of the core

pCPU

A pCPU denotes aphysical CPU, referring to a logical processor on a system with Hyper-Threading(HT) enabled; otherwise, it refers to a processor core.

Virtual Socket 和 VirtualCore

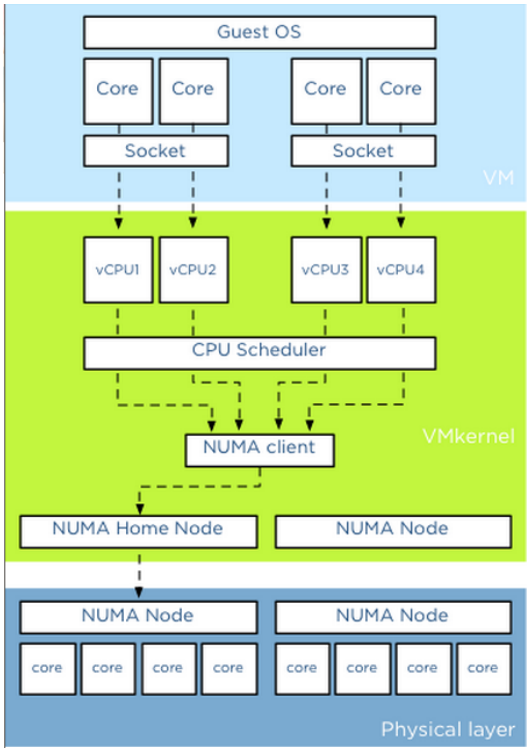
在配置 VM 时设置。运行在 VM 上的操作系统读取该设置。

vCPU

The virtual machine vCPUconfiguration is the sum of number of cores x number of sockets. 比如以一个 VM 有 2 个 Virtual Socket 为例，每个 virtual socket 都有 2 个 Virtual Core。那么这个 VM 一共可用的 vCPU 个数 = number of cores x number of sockets = 2 *2 =4

VM 可以设置的 vCPU 个数，不能超过 ESXi 主机上 Logical CPU 的实际个数

下图揭示了上面一些术语之间的关系。该图摘自 frank denneman 博客



上图分三层，他们分别是是 VM 层，VMKernel 层和物理层。对于物理服务器而言，所有的 CPU 资源都分配给单独的操作系统和上面运行的应用。应用将请求先发送给操作系统，然后操作系统调度物理的 CPU 资源。

在虚拟化平台中，在 VM 层和物理层之间加入了 VMkernel 层，从而允许所有的 VM 共享物理层的资源。VM 上的应用将请求发送给 VM 上的操作系统，然后操纵系统调度 Virtual CPU 资源（操作系统认为 Virtual CPU 和物理 CPU 是一样的），然后 VMkernel 层对多个物理 CPU Core 进行资源调度，从而满足 Virtual CPU 的需要。在虚拟化平台中 OS CPU Scheduler 和 Hyperviisor CPU Scheduler 都在各自的领域内进行资源调度。

关于 NUMA 我们会另文阐述.

二实例：

我们来假设一个主机有 2 个 socket，每个 socket 有 4 个 core。主频 2.4G MHZ 那么一共可用的资源是 2*4*2.4G= 19.2G MHZ

假设主机上运行了三个 VM，VM1 和 VM2 设置为 1socket*1core，VM3 设置为 1socket*2core。那么 VM1 和 VM2 分别有 1 个 vCPU，而 VM3 有 2 个 vCPU。假设其他设置为缺省设置。

那么三个 VM 获得该主机 CPU 资源分配如下：
VM1：25%； VM2：25%； VM3:50%
假设运行在 VM3 上的应用支持多线程，那么该应用可以充分利用到所非配的 CPU 资源。2vCPU 的设置是合适的。

假设运行在 VM3 上的应用不支持多线程，该应用根本无法同时使用利用 2 个 vCPU. 与此同时，VMkernal 层的 CPU Scheduler 必须等待物理层中两个空闲的 pCPU，才开始资源调配来满足 2 个 vCPU 的需要。在仅有 2vCPU 的情况下，对该 VM 的性能不会有太大负面影响。但如果分配 4vCPU 或者更多，这种资源调度上的负担有可能会对该 VM 上运行的应用有很大负面影响。

三 vCPU Sizing 步骤：

以下内容参考了 david Davis 的文章。假如我们要创建一个 VM，以下几步可以帮助确定合适的 vCPU 数目

1 了解应用并设置初始值

该应用是否是关键应用，是否有 Service Level Agreement.
一定要对运行在虚拟机上的应用是否支持多线程深入了解。咨询应用的提供商是否支持多线程和 SMP（Symmetricmulti-processing）。
参考该应用在物理服务器上运行时所需要的 CPU 个数。如果没有参照信息，可设置 1vCPU 作为初始值，然后密切观测资源使用情况。

2 观测资源使用情况

确定一个时间段，观测该虚拟机的资源使用情况。时间段取决于应用的特点和要求，可以是数天，甚至数周。不仅观测该 VM 的 CPU 使用率，而且观测在操作系统内该应用对 CPU 的占用率。
特别要区分 CPU 使用率平均值和 CPU 使用率峰值。假如分配有 4 个 vCPU，可是运行在该 VM 上的应用

的 CPU 使用峰值是 25%， 也就是仅仅能最多使用 25%的全部 CPU 资源，说明该应用是单线程的，仅能够使用一个 vCPU （4 * 25% = 1 ）。
如果平均值小于 38%，而峰值小于 45%，考虑减少 vCPU 数目
如果平均值大于 75%，而峰值大于 90%，考虑增加 vCPU 数目
3 更改 vCPU 数目并观测结果
每次的改动尽量少，如果可能需要 4vCPU,先设置 2vCPU 在观测性能是否可以接受。

参考：
http://frankdenneman.nl/2013/09/18/vcpu-configuration-performance-impact-between-virtual-sockets-and-virtual-cores/
http://datacenterpost.com/2012/03/vcpu-sizing-considerations.html
http://www.vmware.com/files/pdf/techpaper/VMware-vSphere-CPU-Sched-Perf.pdf
http://vpivot.com/2010/04/30/how-many-virtual-cpus-per-vm/

深入 Hadoop 节点部署的策略

作者：vBigData 来自：http://vbigdata.blog.51cto.com/7526470/1316214

简介

每个 Hadoop 集群都包括不止一个节点。这些节点可以按功能分成几个组，比如 master 组，worker 组，client 组等等。每个组里面的节点都有不同的资源要求，比如 CPU，内存，存储。

在虚拟数据中心（DataCenter）中有一个或多个虚拟集群（Cluster），每个虚拟集群中有一个或多个主机（Host）。

节点部署是在保证每个节点资源需求的前提下，按照一定策略将 Hadoop 集群中的所有节点部署到物理主机上。

定义 Hadoop 集群

Hadoop 集群可以通过 UI 来定义，也可以通过 json 格式的文件来定义。这里以 json 文件为例来说明有哪些参数可以使用。

```
1 {
2 "nodeGroups" : [
3 {
4 "name": "master",
5 "roles": [
6 "hadoop_namenode",
7 "hadoop_jobtracker"
8 ],
9 "instanceNum": 1,
10 "instanceType": "LARGE",
11 "cpuNum": 2,
12 "memCapacityMB":4096,
13 "storage": {
14 "type": "SHARED",
15 "sizeGB": 20
16 },
17 "haFlag":"on",
18 "rpNames": [
19 "rp1"
20 ]
21 },
```

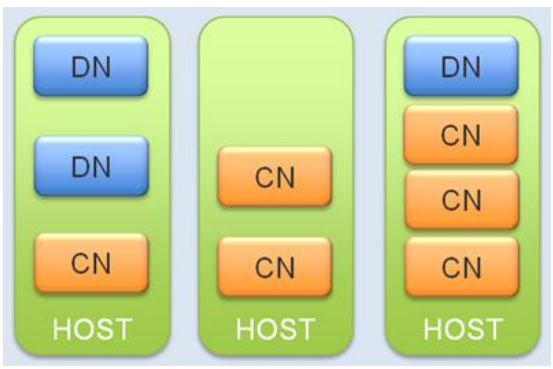
上面的文件片段定义了 master 节点组（第 2 行“name”），其中包括一个节点（第 9 行“instanceNum”），每个节点需要 2 个虚拟 CPU（第 11 行“cpuNum”），4096MB 的内存（第 12 行“memCapacityMB”），20GB 共享的存储空间（第 13-16 行）。

简单的节点部署策略

节点部署的策略有多种，先介绍最简单的一种。因为在一般情况下，Hadoop 节点需要占用的磁盘空间是最紧张的资源，所以最基本的节点部署策略是按照磁盘空间的大小来考虑的。首先将所有 Hadoop 节点按照磁盘空间的要求从大到小进行排序，再将可用的 Host 按照剩余磁盘空间从大到小进行排序。第一步将磁盘空间需求最大的 Hadoop 节点部署到剩余磁盘空间最大的主机上去，第二步将磁盘空间需求次大的 Hadoop 节点部署到剩余磁盘空间次大的主机上去，以此类推。当然，如果后面碰到主机的剩余磁盘空间不够部署某个节点的话，前面拥有较大剩余磁盘空间的主机就会被考虑。这种策略的目的是尽量利用所有可用的主机来部署 Hadoop 节点，这样每个节点都有更多的资源可供使用。

数据和计算分离的节点部署策略

随着 Hadoop 的发展，为了提供更好的弹性和实现真正的多租户，Hadoop 数据节点（DataNode）和计算节点（ComputeNode）分开部署的方式开始流行起来。在这种模式下，为了提供最好的性能，需要把数据节点和计算节点尽可能的部署在相同的主机上。而前面介绍的简单节点部署策略由于没有考虑这种要求，会出现下面这种不好的部署方案。



图中有 2 个数据节点（DN）被放在了一个主机上，而有 2 个计算节点（CN）被放在了没有数据节点的主机上。

为了实现数据节点和计算节点分离下的最优部署，需要在定义 Hadoop 集群时指定一些参数。

我们先来看数据节点的定义：

```
22 {
23 "name": "data",
24 "roles": [
25 "hadoop_datanode"
26 ],
27 "instanceNum": 3,
28 "instanceType": "MEDIUM",
29 "cpuNum": 2,
30 "memCapacityMB":2048,
31 "storage": {
32 "type": "LOCAL",
33 "sizeGB": 50,
34 "dsNames4Data": ["DSLOCALSSD"],
35 "dsNames4System": ["DSNDFS"]
36 }
37 "placementPolicies": {
38 "instancePerHost": 1,
39 "groupRacks": {
40 "type": "ROUNDROBIN",
41 "racks": ["rack1", "rack2", "rack3"]
42 }
```

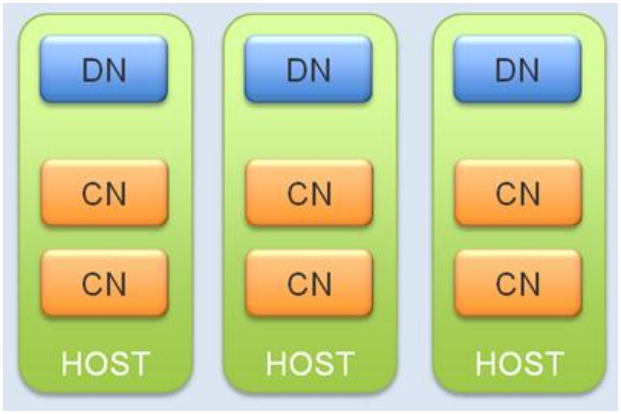
数据节点在每个主机上只部署 1 个（第 38 行“instancePerHost”：1），一共需要 3 个数据节点（第 27 行“instanceNum”：3）。

再看计算节点的定义：

```
45 {
46 "name": "compute",
47 "roles": [
48 "hadoop_tasktracker"
49 ],
50 "instanceNum": 6,
51 "instanceType": "SMALL",
52 "cpuNum": 2,
53 "memCapacityMB":2048,
54 "storage": {
55 "type": "LOCAL",
56 "sizeGB": 10
57 }
58 "placementPolicies": {
59 "instancePerHost": 2,
60 "groupAssociations": [{
61 "reference": "data",
62 "type": "STRICT"
63 }]
64 }
65 },
```

计算节点在每个主机上部署 2 个（第 59 行“instancePerHost”：2），并且必须部署在有数据节点的主机上（第 60 行“groupAssociations”和第 61 行“reference”：“data”）。

在这样的定义下，Hadoop 集群部署后将是下面这种理想的状态：



3 个数据节点（DN）被分别部署在三台主机上，同时每个主机上还部署了 2 个计算节点（CN）。

机架（Rack）感知的节点部署策略

对于大规模的 Hadoop 集群，会在数据中心内使用多个机架（Rack）。通过指定机架名称，可以把特定的 Hadoop 节点组部署在指定的机架上。这种部署方式，可以在某个机架出现问题的时候，让 Hadoop 集群能够在正常工作的机架上持续运行。

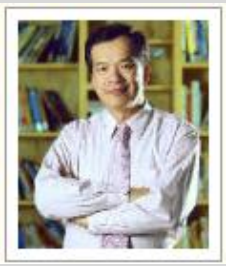
如有任何问题，您可以发邮件至 bigdata_apac@vmware.com

关于 vSphere Big Data Extensions:

VMware vSphere Big Data Extensions（简称 BDE）基于 vSphere 平台支持大数据和 Apache Hadoop 作业。BDE 以开源 Serengeti 项目为基础，为企业级用户提供一系列整合的管理工具，通过在 vSphere 上虚拟化 Apache Hadoop，帮助用户在基础设施上实现灵活、弹性、安全和快捷的大数据部署、运行和管理工作。

Android 架构:认识简法设计与 EIT 软件造形

作者：高焕堂 来自：http://gaoht.blog.51cto.com/7649449/1287837



Android 开发专家 高焕堂
Android 专家顾问。台湾 Android 论坛主席 ,现任亚太地区 Android 技术大会主席，台湾 Android 领域框架开发联盟总架构师。发表 100 多篇 Android 核心技术文章 ,出版了 6 本 Android 专业技术书籍。专精于 Android 核心框架及核心服务程序开发！

一、简单法则(减法设计)

没有简单，我们就无法理解复杂；欲面对复杂，只能简单。

大画家毕加索曾说过，绘画艺术是减法的过程(process of elimination)，画出的作品具有<简单性>。美国麻省理工大学的工业设计教授 John Maeda 也说道：

“减法设计可以是一种富有创意的方式，来看待由设计所推动的世界。简单可以满足人们心灵中想解决谜团、追寻正确完形的自然渴望。”

例如，智慧城市等的设计是复杂的，在垂直面向，切分为上层、中层和实践层；在水平面向，切分为不同的业务区块(Business Area)；然后分而治之。这些都是减法设计思维的表现。幅员愈广大的国家，整体设计规模就愈大(例如数据量，网络带宽等)，其智慧城市的”中层架构的减法设计”(简称中层设计)就愈重要。其如同树木一般，长得愈高大的树，其中层的树干部分就必须简单而强壮。

二、减法设计是战略，造形是会赢的战术

基于减法设计策(战)略，当你无法删除身外物(如软件或太阳星系)的复杂关系时；不须要难过，因为它能简化你脑海里的想象，提升你内心面对复杂事物的能力。造形(Form)则是艺术家、设计师的减法设计呈现，它具有简单特质，提升了人们理解和掌握外界复杂多变的能力。

兹回忆，于十七世纪中，牛顿提出了简单公式(即造形)：F=ma；让人们能轻易理解物体运动的复杂<关系>。再如，于二十世纪初，爱因斯坦发表了简单公式：E=MC 平方；让人们能理解复杂的质量、能量与光速之间的复杂关系。同样地，著名的信息化软件专家 Fred Brooks(“人月神话”一书作者)在 40 年前就说道：

”软件的复杂性是本质性的，并非表像而已。”

(The complexity of software is an essential property, not an accidental one.)

于是，高焕堂老师提出简单的 EIT 软件造形；让人们能理解智慧城市幕后信息化软件体系里的复杂关系。有了中层设计造形的<简单性>，人们就很容易理解智慧城市的中层架构(如同树干结构)的一致性，如何有效支撑上层设计里的繁杂业务功能(如同树叶)了；又能体现、落实于底层快速演化的信息化(IT)新技术上。

三、EIT 软件造形是什么?

1、EIT 是一种基本的结构(Structure)，一种概念(Concept)；我们称它为” EIT 造形(Form)”。

2、同样地，圆形、椭圆等是基本的几何结构，都是概念；称为阿基米得几何造形。

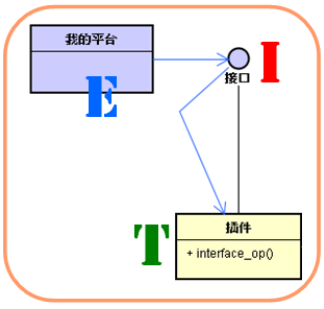


图-1 EIT 造形

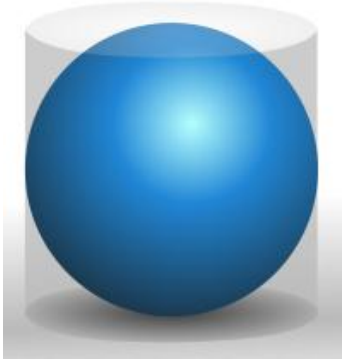


图-2 几何造形

3、同样地，原子(Atom)是基本的物质结构，也是概念；是物理学的基本造形。

4、同样地，唐诗的七言绝句是基本的文章结构，也是概念；是诗词的基本造形。



图-3 原子造形



图-4 唐诗造形

5、同样的，集装箱(Container)是物流航运的货物结构，也是概念；是物流的基本造形。



图-5 集装箱造形

6、同样地，软件的类别(Class)是基本的软件结构，也是概念；是软件的基本造形。

7、同样地，枫叶、樱花都是一种基本结构，也是概念；是赏枫叶、赏花者心中的基本造形。

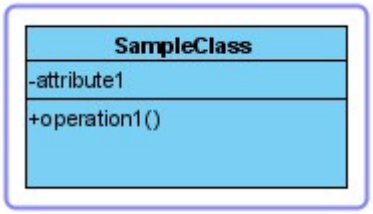


图-6 软件类别造形



图-7 枫叶造形

四、EIT 造形的用途

1、未来的变化和复杂性是本质性(Essential)的，人们无法删除它。于是想办法提升人们管理复杂的能力。

2、著名软件专家 Fred Brooks(“人月神话”一书作者)在 40 年前就说道：“软件的复杂性是本质性的，并非表象而已。”(The complexity of software is an essential property, not an accidental one.)

3、于是，也必须想办法提升人们面对软件复杂性的能力。

4、试想，当你去观察太阳星系时，如果你脑海里没有阿基米得(Archimedes)几何学的”椭圆”概念或造形，你会觉得各星体运行的轨迹非常复杂。这个复杂性是本质的，无法删除，怎么办呢?

5、等到你学了几何，脑海里有了”椭圆”概念或造形之后，再去看太阳系，你会发现只不过是 9 个椭圆形轨道而已，非常简单。

6、此时，你并没有去删除你身外星球的复杂关系；但却简化了脑海里的想象，提升你内心面对复杂事物的能力。

7、同样地，当你去观察 Android 的多层框架体系时，如果你脑海里没有”EIT 概念或造形”，你会觉得 Android 平台软件非常复杂。这个复杂性是本质的，无法删除，怎么办呢?

8、等到你学了 EIT 造形、脑海里有了 EIT 概念之后，再去看 Android 平台软件，你会发现只不过是一 堆 EIT 造形的有机(Organic)组合而已，非常简单。例如，在 <HTML5+Android 平台>的架构里，就是两层 EIT 造形的简单组合而已，如下图所示：

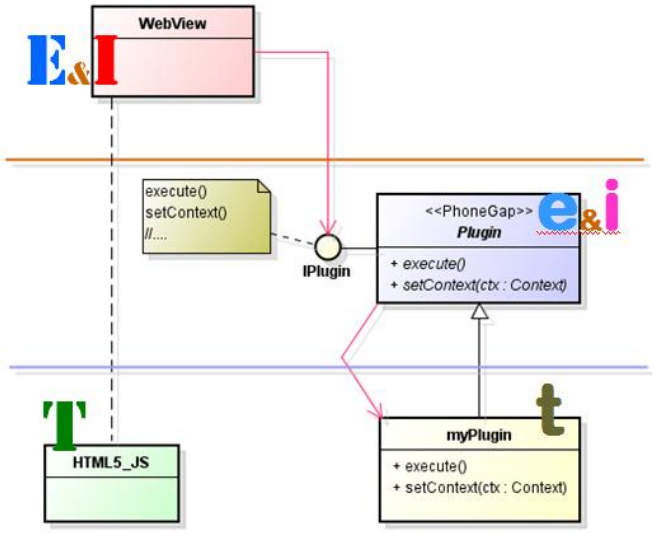


图-8 复杂外貌来自于 EIT 造形的简单组合

所以，EIT 造形的主要用途是：提升内在能力、管理外在变化和复杂。

查看相关文章：
http://gaoht.blog.51cto.com/
查看他的视频课程：
http://edu.51cto.com/course/course_id-539.html

Zend Framework 1.12 中接受 ajax 传来的 json 数据

作者：cnn237111 来自：http://cnn237111.blog.51cto.com/2359144/1310414

首先要知道我们常所说的 json 对象其实是一个误说，实际上指的是 javascript 对象。javascript 语言支持按照 json 格式的方式创建对象。转一篇文章，说的非常完善。
http://www.cnblogs.com/TomXu/archive/2012/01/11/2311956.html

因此要传递 json 数据，通常都是传递一串符合 json 格式规范的字符串而已，并且将 contentType 设置成"application/json",如果传递的是 javascript 对象（即所谓的 json 对象），则会把传递过去的对象转换成 key=value&key1=value1 的格式。由于这种转换后，字符串就不符合 json 格式，因此后台的程序无法把它当做 json 对象来处理。

我们可以用程序来判断是否是合法的 json 字符串，以 PHP 语言，在 Zend Framework 框架下为例。
假设前端的 javascript 代码如下：

```
<script type="text/javascript">
   $(function(){
      jsondata={"value":["w2",3]};
        $("#btnPost").click(function(){
        var options = {
            type: 'POST',
            url: "anypage",
            data: jsondata,
            success: function (result) {
              alert(result);
            },
            dataType: "html",
            contentType:"application/json"
            };
        $.ajax(options);
      });
            });
</script>
```

传递的是一个 jsondata，这是一个 javascript 对象，那么再传递到后台判断是否是符合 json 格式。

```
$contentType = $this->getRequest()->getHeader('Content-Type');
if(strstr($contentType, 'application/json'))
{
  $rawBody=$this->getRequest()->getRawBody();
  json_decode($rawBody);
  echo (json_last_error() == JSON_ERROR_NONE)? "it's valid json data":"Not a valid json data";
}
```

通过这段代码，我们点击按钮后，会弹出"Not a valid json data"对话框。它实际上传过去的字符串是 value%5B%5D=w2&value%5B%5D=3，后台无法处理按 json 格式处理这样的数据，硬是要这样处理的话，
一种方法是去掉 contentType:"application/json"，使之成为默认的 application/x-www-form-urlencoded，然后通过 getParam 的方法去获得值，比如下面的代码
echo $this->getRequest()->getParam("value")[0];
第二种方法是调用方法 parse_str(),直接解析出变量值。注意：传递 contentType:"application/json"这种类型的数据，通常都是采用 request 的 getRawBody()方法获取。

```
$rawBody=$this->getRequest()->getRawBody();
parse_str($rawBody);
echo $value[0];
```

那这就有点偏离本文的目的：如何介绍处理 json。其实通过上面的一些铺垫，可以知道大概，首先，得传递 Json 字符串到后台，后台要获得这个正真的字符串后，转换成 PHP 自己的对象。在 PHP 环境中，要传递 contentType:"application/json"这种类型的数据，通常都是采用 request 的 getRawBody()方法，来获取原始的 post 请求的信息。示例代码如下：

```
<script type="text/javascript">
$(function(){
    jsondata={"value1":["w2",3],"value2":990};//假设传递的是这样的数据
    $("#btnPost").click(function(){
      var options = {
      type: 'POST',
      url: "anyAction",
      data: JSON.stringify(jsondata),
      success: function (result) {
        alert(result);
      },
      contentType:"application/json",
      dataType: "html"
    };
    $.ajax(options);
    });
});
</script>
```

后台代码为：

```
public function anyAction()
{
  $contentType = $this->getRequest()->getHeader('Content-Type');
  if(strstr($contentType, 'application/json'))
  {
      $rawBody=$this->getRequest()->getRawBody();
      $json=Zend_Json::decode($rawBody);//此处$json就是一个数组，可以把它再次转换成对象
      echo $json["value1"][0],$json["value1"][1],$json["value2"],"\r\n";
      $o=(object)$json;
      echo $o->value1[0],$o->value1[1],$o->value2,"\r\n";//转换成对象
  }
  $this->getHelper("layout")->disableLayout();
  $this->getHelper("viewRenderer")->setNoRender(true);
}
```

上述的代码，演示了 json 字符串传递到 php 后台，转换成数组，或者对象的方法。仅供参考。

Tomcat 完美实现在线商城应用部署

作者：z 永 来自：http://pangge.blog.51cto.com/6013757/1305100

Tomcat 是一个小型的轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好 Apache 服务器，可利用它响应对 HTML 页面的访问请求。实际上 Tomcat 部分是 Apache 服务器的扩展，但它是独立运行的，所以当你运行 tomcat 时，它实际上作为一个与 Apache 独立的进程单独运行的。

Tomcat 很受广大程序员的喜欢，因为它运行时占用的系统资源小，扩展性好，支持负载平衡与邮件服务等开发应用系统常用的功能；而且它还在不断的改进和完善中，任何一个感兴趣的程序员都可以更改它或在其中加入新的功能。

环境介绍：

系统版本：CentOS 6.4_x86_64

mysql 版本： mysql-server.x86_64 0:5.1.66-2.el6_3

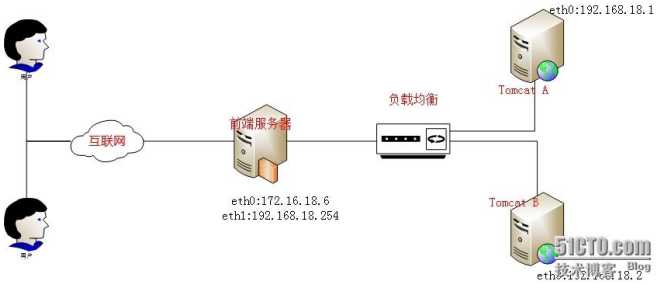
Apache 版本：httpd-2.2.15-26.el6

Tomcat 版本：apache-tomcat-7.0.42

JDK 版本： jdk-7u40-linux-x64

在线商城系统：shopxx-3.0Beta（可下载）

拓扑展示：



安装配置过程：

1、配置 IP 地址

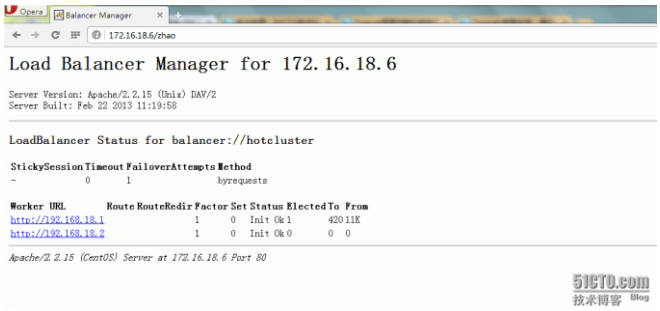
```
############前端服务器[root@node ~]#: ##############
ifconfig eth0 172.16.18.6
ifconfig eth1 192.168.18.254/24
############Tomcat A[root@node1~]#: ################
ifconfig eth0 192.168.18.1/24
route add default gw 192.168.18.254
############Tomcat B[root@node2~]#:#################
ifconfig eth0 192.168.18.2/24
route add default gw 192.168.18.254
```

2、安装配置后端服务器

```
##########TomcatA服务器: ##############
    #########安装JDK: ##############
[root@node1~]# rpm -ivh jdk-7u9-linux-x64.rpm
[root@node1~]# vim /etc/profile.d/java.sh    #编辑执行脚本
 export JAVA_HOME=/usr/java/latest
 export PATH=$JAVA_HOME/bin:$PATH
[root@node1~]# . /etc/profile.d/java.sh     #读取执行脚本
[root@node1~]# java -version                #查看安装信息
javaversion "1.7.0_09"   #版本信息
Java(TM)SE Runtime Environment (build 1.7.0_09-b05)    #运行时环境的版本
JavaHotSpot(TM) 64-Bit Server VM (build 23.5-b02, mixed mode)
     #########安装Tomcat#############
[root@node1~]# tar xf apache-tomcat-7.0.42.tar.gz -C /usr/local/
[root@node1~]# cd /usr/local/
[root@node1local]# ln -sv apache-tomcat-7.0.42/ tomcat
[root@node1tomcat]# vim /etc/profile.d/tomcat.sh      #定义启动环境变量
exportCATALINA_HOME=/usr/local/tomcat
exportPATH=$CATALINA_HOME/bin:$PATH
[root@node1tomcat]# . /etc/profile.d/tomcat.sh       #读取脚本
[root@node1~]# catalina.sh version    #查看安装信息
UsingCATALINA_BASE:   /usr/local/tomcat
UsingCATALINA_HOME:   /usr/local/tomcat
UsingCATALINA_TMPDIR: /usr/local/tomcat/temp
UsingJRE_HOME:        /usr/java/latest
UsingCLASSPATH:      /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Serverversion: Apache Tomcat/7.0.42
Serverbuilt:   Jul 2 2013 08:57:41
Servernumber:  7.0.42.0
OSName:        Linux
OSVersion:     2.6.32-358.el6.x86_64
Architecture:  amd64
JVMVersion:    1.7.0_09-b05
JVMVendor:     Oracle Corporation
############启动##################
[root@node1~]# catalina.sh start
############查看启动端口###########
[root@node1~]# ss -tanl
State      Recv-Q Send-Q                    Local Address:Port                      Peer
Address:Port
LISTEN     0      128                                   :::111
:::*
LISTEN     0      128                                    *:111
*:*
LISTEN     0      100                                   :::8080
:::*
LISTEN     0      128                                   :::49906
:::*
LISTEN     0      128                                    *:56818
*:*
LISTEN     0      128                                   :::22
:::*
LISTEN     0      128                                    *:22
*:*
LISTEN     0      128                             127.0.0.1:631
*:*
LISTEN     0      128                                   ::1:631
:::*
 LISTEN     0      100                                           ::1:25
 :::*
 LISTEN     0      100                             127.0.0.1:25
 *:*
 LISTEN     0      128                             127.0.0.1:6010
 *:*
 LISTEN     0      128                                   ::1:6010
 :::*
 LISTEN     0      128                             127.0.0.1:6011
 *:*
 LISTEN     0      128                                   ::1:6011
 :::*
 LISTEN     0      128                             127.0.0.1:6012
 *:*
 LISTEN     0      128                                   ::1:6012
 :::*
 LISTEN     0      1                        ::ffff:127.0.0.1:8005
 :::*
 LISTEN     0      100                                   :::8009
 :::*
 ###解析：tomcat默认监听在8080端口上，启动监听端口为8005,8009####
 ##########Tomcat B服务器上同样安装，这里就不再重复演示了########
 ##########更改监听端口############
 [root@node1conf]# vim /usr/local/tomcat/conf/server.xml
     <Connector port="80"protocol="HTTP/1.1"            #70行
                connectionTimeout="20000"
                redirectPort="8443"/>
 ##########重新启动服务############
 [root@node1conf]# catalina.sh stop
 [root@node1conf]# catalina.sh start
 [root@node1conf]# scp server.xml 192.168.18.2:/usr/local/tomcat/conf/
 [root@node2local]# catalina.sh stop
 [root@node2local]# catalina.sh start
```

3、配置前端服务器实现负载均衡

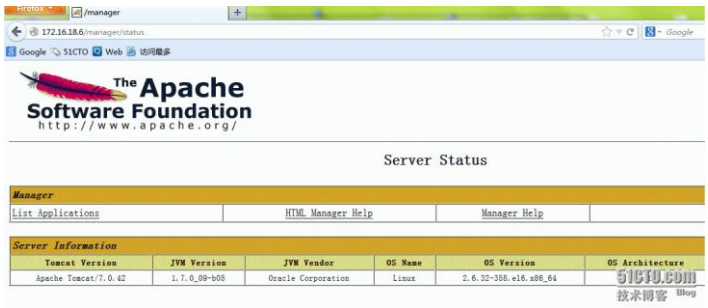
```
##########安装Apache###################
[root@node~]# yum -y install httpd
[root@node~]# service httpd start
##########配置基于mod_proxy代理，负载均衡########
[root@node~]# cd /etc/httpd/conf.d/
[root@nodeconf.d]# vim mod_proxy_tomcat.conf
ProxyViaOn
ProxyRequestsOff      #关闭正向代理
ProxyPreserveHostOff
<Proxybalancer://hotcluster>     #定义代理的后端服务器
BalancerMember  http://192.168.18.39:80 loadfactor=1        #指定后端服务器地址及端口
BalancerMember  http://192.168.18.36:80 loadfactor=1
ProxySet  lbmethod=byrequests
</Proxy>
<Location/zhao>
  SetHandler balancer-manager
  Proxypass !
  Order Deny,Allow
  Allow from all
</Location>
ProxyPass/ balancer://hotcluster/
#stickysession=JSESSIONID|jsessionidnofailover=On
ProxyPassReverse/ balancer://hotcluster/
<Proxy*>
  Order Allow,Deny
  Allow From all
</Proxy>
<Location  / >
  Order Allow,Deny
  Allow From all
</Location>
############测试语法的正确性##############
[root@nodeconf.d]# service httpd configtest
##########重启服务######################
[root@nodeconf.d]# service httpd restart
```



4、查看链接状况开启 tomcat 管理功能：

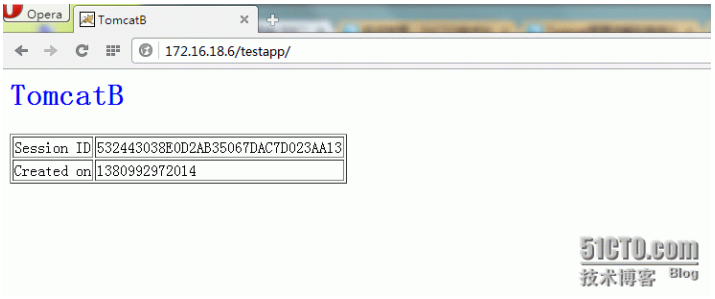
```
##注意：由于两台后端服务器可能都会用到，所以两边都要创建用户并且保持一致##
#########更改用户管理配置文件######
[root@node1conf]# vim /usr/local/tomcat/conf/tomcat-users.xml
##内容：##
  <rolerolename="manager-gui"/>
  <role rolename="admin-gui"/>
  <user username="tomcat"password="tomcat" roles="manager-gui,admin-gui"/>
#########复制到Tomcat B############
[root@node1conf]# scp tomcat-users.xml 192.168.18.2:/usr/local/tomcat/conf/
##########重新启动服务#############
[root@node1conf]# catalina.sh stop
[root@node1conf]# catalina.sh start
[root@node2local]# catalina.sh stop
[root@node2local]# catalina.sh start
```

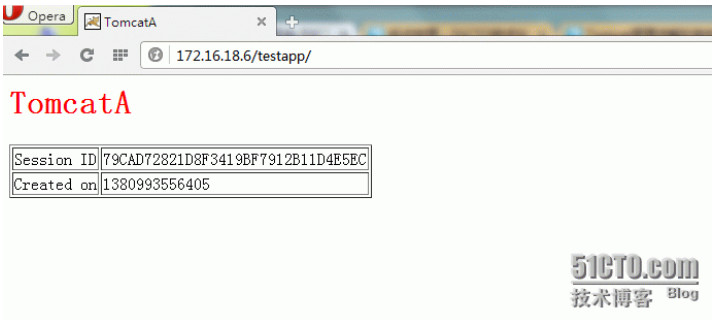




5、简单部署主页查看 session 值：

```
##########部署虚拟目录和部署主页#############
        #########  Tomcat A  ###########
[root@node1 tomcat]#  mkdir webapps/testapp/WEB-INF/{lib,classes} -pv
[root@node1 tomcat]# vim webapps/testapp/index.jsp
###内容###
<%@ page language="java" %>
<html>
  <head><title>TomcatA</title></head>
  <body>
    <h1><font color="red">TomcatA </font></h1>
    <table align="centre" border="1">
      <tr>
        <td>Session ID</td>
    <% session.setAttribute("abc","abc"); %>
        <td><%= session.getId() %></td>
      </tr>
      <tr>
        <td>Created on</td>
        <td><%= session.getCreationTime() %></td>
     </tr>
    </table>
  </body>
</html>
[root@node1 tomcat]# catalina.sh stop
[root@node1 tomcat]# catalina.sh start
          ############ Tomcat B ############
[root@node2 tomcat]#  mkdir webapps/testapp/WEB-INF/{lib,classes} -pv
[root@node2 tomcat]# vim webapps/testapp/index.jsp
###内容###
<%@ page language="java" %>
<html>
  <head><title>TomcatB</title></head>
  <body>
    <h1><font color="blue">TomcatB </font></h1>
    <table align="centre" border="1">
      <tr>
        <td>Session ID</td>
    <% session.setAttribute("abc","abc"); %>
        <td><%= session.getId() %></td>
      </tr>
      <tr>
        <td>Created on</td>
        <td><%= session.getCreationTime() %></td>
     </tr>
    </table>
  </body>
</html>
[root@node2 webapps]# catalina.sh stop
[root@node2 webapps]# catalina.sh start
```





注释：定义之后每次刷新页面都会更改 session ID，也就是说可以保证负载均衡但是无法保证 session ID 不变。

6、基于内存复制实现 session 共享集群：

```
############ Tomcat A ############
[root@node1 ~]# cd /usr/local/tomcat/conf/
[root@node1 conf]# nano server.xml      #由于以下内容有缩进格式建议使用此编辑器
##添加在/host和/engine之间: ##
        <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"
                 channelSendOptions="8">
          <Manager className="org.apache.catalina.ha.session.DeltaManager"
                   expireSessionsOnShutdown="false"
                   notifyListenersOnReplication="true"/>
          <Channel className="org.apache.catalina.tribes.group.GroupChannel">
            <Membership className="org.apache.catalina.tribes.membership.McastService"
                        address="228.0.0.4"
                        port="45564"
                        frequency="500"
                        dropTime="3000"/>
            <Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"
                      address="192.168.18.1"            #自己的地址
                      port="4000"
                      autoBind="100"
                      selectorTimeout="5000"
                      maxThreads="6"/>
            <Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
              <Transport
className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>
            </Sender>
            <Interceptor
className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>
            <Interceptor
className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>
          </Channel>
          <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"
                 filter=""/>
          <Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>
          <Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"
                    tempDir="/tmp/war-temp/"
                    deployDir="/tmp/war-deploy/"
                    watchDir="/tmp/war-listen/"
                    watchEnabled="false"/>
          <ClusterListener
className="org.apache.catalina.ha.session.JvmRouteSessionIDBinderListener"/>
          <ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>
        </Cluster>
#停止tomcat##
[root@node1 ~]# catalina.sh stop
[root@node1 conf]# cp web.xml ../webapps/testapp/WEB-INF/
[root@node1 conf]# vim ../webapps/testapp/WEB-INF/web.xml
添加如下字段:
  <distributable />            #表示这个应用程序的会话将实现分布式
[root@node1 conf]# scp ../webapps/testapp/WEB-INF/web.xml
192.168.18.2:/usr/local/tomcat/webapps/testapp/WEB-INF/
[root@node1 conf]# scp server.xml 192.168.18.2:/usr/local/tomcat/conf/
[root@node1 conf]# catalina.sh start
############ Tomcat B ############
[root@node1 ~]# catalina.sh stop
[root@node1 conf]# catalina.sh start
```



总结：我们发现无论如何刷新，session id 的信息都不会再改变。至此我们完成了使用 mod_proxy 实现负载均衡和利用 tomcat 集群实现 session 共享。下面我们来部署商店的应用。

7、在线商店的应用实现

```
############ Tomcat A ############
[root@node1 ~]# unzip shopxx-a5-Beta.zip
[root@node1 ~]# mv shopxx-v3.0-Beta/* /usr/local/tomcat/webapps/shopx
[root@node2 tomcat]# vim webapps/shopxx/WEB-INF/web.xml
<distributable />   #添加此字段
############ Tomcat B ############
[root@node2 ~]# unzip shopxx-a5-Beta.zip
[root@node2 ~]# mv shopxx-v3.0-Beta/* /usr/local/tomcat/webapps/shopx
[root@node2 tomcat]# vim webapps/shopxx/WEB-INF/web.xml
<distributable />   #添加此字段
```





注：安装环境不符合基本环境需要增加内存：方法如下

```
######## Tomcat A ########################
[root@node1 ~]# cd /usr/local/tomcat/bin/
[root@node1 bin]# vim catalina.sh
JAVA_OPTS="$JAVA_OPTS -Xmx256M"      //237行
注释：修改java虚拟机的内存需要在启动脚本里修改；  -Xmx:定义java虚拟机使用最大内存。其中M：指的是MB，不指定则为KB
######### Tomcat B 服务器也需要更改#######
#########重新启动服务######################
```



更改完之后下一步：无法进入到安装页面会重新回到进入页面；

原因：因为安装时后端可能会出现负载均衡转移所以导致无法进入安装页面，只能使用本地安装

```
######切换至前端服务器##########
[root@node ~]# yum -y install firefox
[root@node ~]# firefox &
```

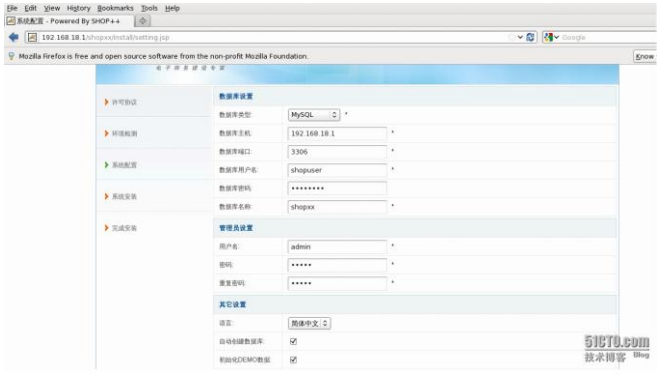
安装商城应用

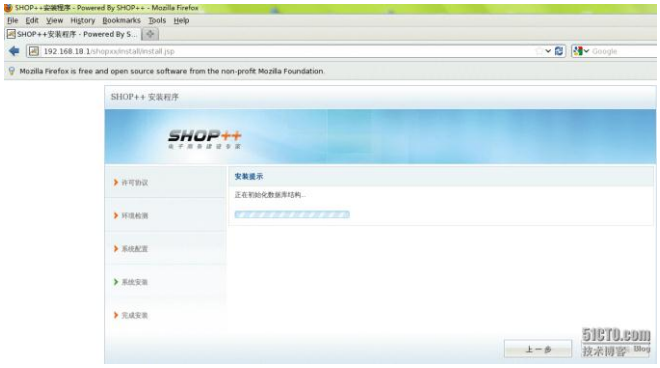




安装 mysql 服务器，并授权一个用户用于登录安装服务；

```
############ Tomcat A ############
[root@node1 shopxx]# yum -y install mysql-server
[root@node1 shopxx]# service mysqld start
[root@node1 shopxx]# mysql
mysql> grant all on *.* to 'shopuser'@'192.168.%.%' identified by 'shoppass';
mysql> flush privileges;
############ Tomcat B ############
[root@node2 shopxx]# yum -y install mysql-server
[root@node2 shopxx]# service mysqld start
[root@node2 shopxx]# mysql
mysql> grant all on *.* to 'shopuser'@'192.168.%.%' identified by 'shoppass';
mysql> flush privileges;
```







点击完成安装，重启 web 服务删除 install 目录

```
[root@node1 shopxx]# mv install install.bak
[root@node1 shopxx]# catalina.sh stop
[root@node1 shopxx]# catalina.sh start
```

将 Tomcat A 上 shopxx 安装文件复制到 Tomcat B 上

```
[root@node1 webapps]# scp -rp shopxx/* 192.168.18.2:/usr/local/tomcat/webapps/shopxx/
[root@node2 shopxx]# catalina.sh stop
[root@node2 shopxx]# catalina.sh start
```



下面就可以可以在商城系统注册一个帐号，并登录到系统，购买一些衣服等，查看购物车中是否保存已加入到购物车中的衣服等等。



至此，tomcat 基于 session 共享集群实现在线商城应用部署就完美实现了，注意此商城不可做商业用途。

C#设计模式(2)——简单工厂模式

作者：LearningHard 来自：http://learninghard.blog.51cto.com/6146675/1289553

一、引言

这个系列也是自己对设计模式的一些学习笔记,希望对一些初学设计模式的人有所帮助的,在上一个专题中介绍了单例模式,在这个专题中继续为大家介绍一个比较容易理解的模式——简单工厂模式。

二、简单工厂模式的介绍

说到简单工厂，自然的第一个疑问当然就是什么是简单工厂模式了？ 在现实生活中工厂是负责生产产品的,同样在设计模式中,简单工厂模式我们也可以理解为负责生产对象的一个类, 我们平常编程中，当使用"new"关键字创建一个对象时，此时该类就依赖与这个对象，也就是他们之间的耦合度高，当需求变化时，我们就不得不去修改此类的源码，此时我们可以运用面向对象（OO）的很重要的原则去解决这一的问题，该原则就是——封装改变，既然要封装改变，自然也就要找到改变的代码，然后把改变的代码用类来封装，这样的一种思路也就是我们简单工厂模式的实现方式了。下面通过一个现实生活中的例子来引出简单工厂模式。

在外面打工的人，免不了要经常在外面吃饭，当然我们也可以自己在家做饭吃，但是自己做饭吃麻烦，因为又要自己买菜，然而，出去吃饭就完全没有这些麻烦的，我们只需要到餐馆点菜就可以了，买菜的事情就交给餐馆做就可以了，这里餐馆就充当简单工厂的角色，下面让我们看看现实生活中的例子用代码是怎样来表现的。

自己做饭的情况：

```
/// <summary>
    /// 自己做饭的情况
    /// 没有简单工厂之前，客户想吃什么菜只能自己炒的
    /// </summary>
    public class Customer
    {
        /// <summary>
        /// 烧菜方法
        /// </summary>
        /// <param name="type"></param>
        /// <returns></returns>
        public static Food Cook(string type)
        {
            Food food = null;
            // 客户A说：我想吃西红柿炒蛋怎么办？
            // 客户B说：那你就自己烧啊
            // 客户A说： 好吧，那就自己做吧
            if (type.Equals("西红柿炒蛋"))
            {
                food = new TomatoScrambledEggs();
            }
            // 我又想吃土豆肉丝，这个还是得自己做
            // 我觉得自己做好累哦，如果能有人帮我做就好了？
            else if (type.Equals("土豆肉丝"))
            {
                food = new ShreddedPorkWithPotatoes();
            }
            return food;
        }
        static void Main(string[] args)
        {
            // 做西红柿炒蛋
            Food food1 = Cook("西红柿炒蛋");
            food1.Print();
            Food food2 = Cook("土豆肉丝");
            food1.Print();
            Console.Read();
        }
    }
 /// <summary>
    /// 菜抽象类
    /// </summary>
    public abstract class Food
    {
        // 输出点了什么菜
        public abstract void Print();
    }
    /// <summary>
    /// 西红柿炒鸡蛋这道菜
    /// </summary>
    public class TomatoScrambledEggs : Food
    {
        public override void Print()
        {
            Console.WriteLine("一份西红柿炒蛋！");
        }
    }
    /// <summary>
```

自己做饭，如果我们想吃别的菜时，此时就需要去买这种菜和洗菜这些繁琐的操作，有了餐馆（也就是简单工厂）之后，我们就可以把这些操作交给餐馆去做，此时消费者（也就是我们）对菜（也就是具体对象）的依赖关系从直接变成的间接的，这样就是实现了面向对象的另一个原则——降低对象之间的耦合度，下面就具体看看有了餐馆之后的实现代码（即简单工厂的实现）：

```
/// <summary>
    /// 顾客充当客户端，负责调用简单工厂来生产对象
    /// 即客户点菜，厨师（相当于简单工厂）负责烧菜(生产的对象)
    /// </summary>
    class Customer
    {
        static void Main(string[] args)
        {
            // 客户想点一个西红柿炒蛋
            Food food1 = FoodSimpleFactory.CreateFood("西红柿炒蛋");
            food1.Print();
            // 客户想点一个土豆肉丝
            Food food2 = FoodSimpleFactory.CreateFood("土豆肉丝");
            food2.Print();
            Console.Read();
        }
    }
    /// <summary>
    /// 菜抽象类
    /// </summary>
    public abstract class Food
    {
        // 输出点了什么菜
        public abstract void Print();
    }
    /// <summary>
    /// 西红柿炒鸡蛋这道菜
    /// </summary>
    public class TomatoScrambledEggs : Food
    {
        public override void Print()
        {
            Console.WriteLine("一份西红柿炒蛋！ ");
        }
    }
    /// <summary>
    /// 土豆肉丝这道菜
    /// </summary>
    public class ShreddedPorkWithPotatoes : Food
    {
        public override void Print()
        {
            Console.WriteLine("一份土豆肉丝");
        }
    }
    /// <summary>
    /// 简单工厂类，负责 炒菜
    /// </summary>
    public class FoodSimpleFactory
    {
        public static Food CreateFood(string type)
```

三、优点与缺点

看完简单工厂模式的实现之后，你和你的小伙伴们肯定会有这样的疑惑（因为我学习的时候也有）——这样我们只是把变化移到了工厂类中罢了，好像没有变化的问题，因为如果客户想吃其他菜时，此时我们还是需要修改工厂类中的方法（也就是多加 case 语句，没应用简单工厂模式之前，修改的是客户类）。我首先要说：你和你的小伙伴很对，这个就是简单工厂模式的缺点所在（这个缺点后面介绍的工厂方法可以很好地解决），然而，简单工厂模式与之前的实现也有它的优点：

简单工厂模式解决了客户端直接依赖于具体对象的问题，客户端可以消除直接创建对象的责任，而仅仅是消费产品。简单工厂模式实现了对责任的分割。

简单工厂模式也起到了代码复用的作用，因为之前的实现（自己做饭的情况）中，换了一个人同样要去在自己的类中实现做菜的方法，然后有了简单工厂之后，去餐馆吃饭的所有人都不用那么麻烦了，只需要负责消费就可以了。此时简单工厂的烧菜方法就让所有客户共用了。（同时这点也是简单工厂方法的缺点——因为工厂类集中了所有产品创建逻辑，一旦不能正常工作，整个系统都会受到影响，也没什么不好理解的，就如事物都有两面性一样道理）

虽然上面已经介绍了简单工厂模式的缺点，下面还是总结下简单工厂模式的缺点：

工厂类集中了所有产品创建逻辑，一旦不能正常工作，整个系统都会受到影响（通俗地意思就是：一旦餐馆没饭或者关门了，很多不愿意做饭的人就没饭吃了）

系统扩展困难，一旦添加新产品就不得不修改工厂逻辑，这样就会造成工厂逻辑过于复杂。

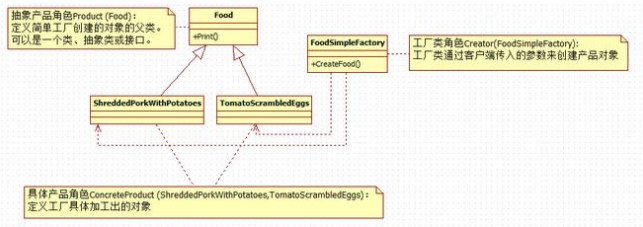
了解了简单工厂模式之后的优缺点之后，我们之后就可以知道简单工厂的应用场景了：

当工厂类负责创建的对象比较少时可以考虑使用简单工厂模式（ ）

客户如果只知道传入工厂类的参数，对于如何创建对象的逻辑不关心时可以考虑使用简单工厂模式

四、简单工厂模式 UML 图

简单工厂模式又叫静态方法模式（因为工厂类都定义了一个静态方法），由一个工厂类根据传入的参数决定创建出哪一种产品类的实例（通俗点表达：通过客户下的订单来负责烧那种菜）。简单工厂模式的 UML 图如下：



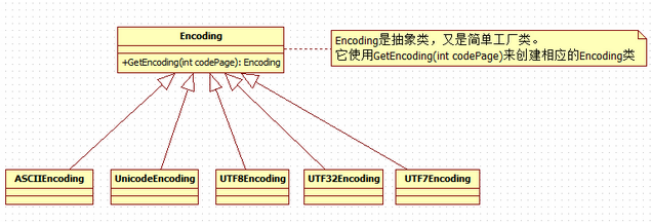
五、.NET 中简单工厂模式的实现

介绍完了简单工厂模式之后，我学习的时候就像：.NET 类库中是否有实现了简单工厂模式的类呢？后面确实有，.NET 中 System.Text.Encoding 类就实现了简单工厂模式，该类中的 GetEncoding(int codepage)就是工厂方法，具体的代码可以通过 Reflector 反编译工具进行查看，下面我也贴出该方

法中部分代码：

```
public static Encoding GetEncoding(int codepage)
{
    Encoding unicode = null;
    if (encodings != null)
    {
        unicode = (Encoding) encodings[codepage];
    }
    if (unicode == null)
    {
        object obj2;
        bool lockTaken = false;
        try
        {
            Monitor.Enter(obj2 = InternalSyncObject, ref lockTaken);
            if (encodings == null)
            {
                encodings = new Hashtable();
            }
            unicode = (Encoding) encodings[codepage];
            if (unicode != null)
            {
                return unicode;
            }
            switch (codepage)
            {
                case 0:
                    unicode = Default;
                    break;
                case 1:
                case 2:
                case 3:
                case 0x2a:
                    throw new
ArgumentException(Environment.GetResourceString("Argument_CodepageNotSupported", new object[]
{ codepage }), "codepage");
                case 0x4b0:
                    unicode = Unicode;
                    break;
                case 0x4b1:
                    unicode = BigEndianUnicode;
                    break;
                case 0x6faf:
                    unicode = Latin1;
                    break;
                case 0xfde9:
                    unicode = UTF8;
                    break;
                case 0x4e4:
                    unicode = new SBCSCodePageEncoding(codepage);
                    break;
                case 0x4e9f:
                    unicode = ASCII;
                    break;
                default:
                    unicode = GetEncodingCodePage(codepage);
                    if (unicode == null)
                    {
                        unicode = GetEncodingRare(codepage);
```

.NET 中 Encoding 的 UML 图为：



Encoding 类中实现的简单工厂模式是简单工厂模式的一种演变，此时简单工厂类由抽象产品角色扮演，然而.NET 中 Encoding 类是如何解决简单工厂模式中存在的问题的呢（即如果新添加一种编码怎么办）？在 GetEncoding 方法里的 switch 函数有如下代码：

```
switch (codepage)
    {
        .......
  default:
                    unicode = GetEncodingCodePage(codepage);
                    if (unicode == null)
                    {
                        unicode = GetEncodingRare(codepage); //当编码很少见时
                    }
                    break;
          ......
      }
```

在 GetEncodingRare 方法里有一些不常用编码的实例化代码，微软正式通过这个方法来解决新增加一种编码的问题。（其实也就是列出所有可能的编码情况），微软之所以以这样的方式来解决这个问题，可能是由于现在编码已经稳定了，添加新编码的可能性比较低，所以在.NET 4.5 仍然未改动这部分代码。

六、总结

到这里，简单工厂模式的介绍都到这里了，后面将介绍工厂方法模式来解决简单工厂模式中存在的问题。

查看相关专题：
C#设计模式详解
http://blog.51cto.com/zt/537

C 语言进阶【文件】常用文件操作函数详解

作者：zyuqiang 来自：http://zhaoyuqiang.blog.51cto.com/6328846/1296902

C 语言学习笔记

之 关于文件操作函数

文件操作是 C 语言的最后篇章了，一起来看一下用 C 语言怎么操作文件

一：文件指针

我们所说的文件不同于一个数据类型，比如一个 test.txt 文档，它有自己的一些属性，比如文件的当前位置、与该文件对应的内存缓冲区地址、文件操作方式等等。这些信息都在专门开辟一定空间的内存中存储着。

在内存中存储着是不错，而且它们还保存在一个结构体类型的变量中，比如：

```
type struct
{
    int fd;//文件号
    int cleft;//缓冲区中剩下的字符
    int mode;//文件操作模式
    char *nextc;//下一个字符位置
    char *buff;//文件缓冲区位置
}FILE;
/*FILE是一个存储文件信息的结构体类型的变量*/
```

在上述代码中可以看到，文件号、文件操作模式等信息都存储在一个结构体中。

这个结构体是系统自己定义的，定义在 stdio.h 头文件中，取名为 FILE，也就是说 FILE 是系统名称，不可改变的，不是自定义。只要程序用到一个文件，系统就为此文件开辟一个如上的结构体变量。需要几个文件，系统就开辟几个这样的结构体。

这些结构体变量不用变量名来标识，而是通过指向结构体类型的指针变量去访问，这就是文件指针。

比如：FILE *fp1,*fp2,*fp3; 当引用文件时，就可以把这个文件的结构体首地址赋给某个文件指针（*fp1）,通过*fp1 就可以调用结构体以找到相关的信息。

一般来说，有几个文件就有几个结构体，有几个结构体就有几个文件指针。

二：文件操作：文件的打开与关闭。

对磁盘文件的操作必须是先打开，然后读写，最后关闭。

打开和关闭扮演着重要的角色，打不开文件也就代表着我们要对文件的操作是徒劳的吧。

1.文件的打开

【格式】fopen(文件名，文件使用方式)

【说明】fopen 库函数，用来打开文件的函数，文件名为文件所在路径，文件使用方式有如下几种：

文件使用方式	含义
“r”（只读）	为输入打开一个文本文件
“w”（只写）	为输出打开一个文本文件
“a”（追加）	想文本文件尾增加数据
“rb”（只读）	为输入打开一个二进制文件
“wb”（只写）	为输出打开一个二进制文件
“ab”（追加）	向二进制文件尾增加数据
“r+”（读写）	为读写打开一个文本文件
“w+”（读写）	为读写打开一个新的文本文件
“a+”（读写）	为读写打开一个文本文件
“rb+”（读写）	为读写打开一个二进制文件
“wb+”（读写）	为读写建立一个新的二进制文件
“ab+”（读写）	为读写打开一个二进制文件
“t”（文本文件）可以省略不写	

总结开来：就是 r 为读 w 为写 b 为二进制文件 a 为追加 +为读写。可以组合使用

【例如】

```
FILE *fp;
fp=fopen("e:\\test.txt","r");
```

含义是：在当前目录下，打开 e 盘下的 test.txt 文档，只允许“读”操作，并使 fp 指向该文件。

【返回值】如果不能成功打开，fopen 将带回一个数字 0 的返回值，这个 0 实际上在 stdio.h 文件中被定义为空指针值 NULL。常用下面方法打开一个文件：

```
if((fp=fopen("e:\\test.txt","rt"))==NULL)
  {
     printf("\nConnot open file!");//如果打开不成功则显示不能打开文件
  getchar();
  exit(1);//关闭所有文件，终止正调用的过程
   }
```

2.文件的关闭

文件的关闭是必不可缺少的操作，如果不关闭文件可能造成数据丢失等后果。

文件的关闭相对来说要简单很多，只是一个执行命令而已。

【格式】fclose(fp);

【返回值】当顺利关闭后，返回 0，如果为非 0 值，则说明关闭有错误。

关闭的过程：先将缓冲区中尚未存盘的数据写盘，然后撤销存放该文件信息的结构体，最后令指向该文件的指针为空。

关闭完成后如果再想打开文件，则应重新打开。

三：文件的读写

常用的文件读写函数有三类：

字符读写函数：fgetc()和 fputc() (getc 和 putc)

字符串读写函数：fgets()和 fputs()

数据块读写函数：fread()和 fwrite()

格式化读写函数：fscanf()和 fprintf()

1. 字符读写函数：fgetc()和 fputc() (getc 和 putc)

字符读写函数是以字符（字节）为单位的读写函数，也就是说每次操作都是读入或写入一个字符。

函数 fgetc 可以写成 getc，而 fputc 可以写成 putc。

（1）fgetc()字符读入函数

【格式】ch=fgetc(pf)

【含义】从 pf 所指定的文件中带回一个字符赋给 ch

【返回值】如果执行 fgetc 读字符时遇到文件结束符，函数返回一个文件结束标志 EOF。（EOF 在 stdio.h 中定义为-1）

【程序实例】

```
#include<stdio.h>
#include<stdlib.h>//exit在此头文件中
main()
{
   FILE *fp;//定义文件指针
   char ch;
   /*判断文件打开是否成功*/
   /*打开方式为rt,读文件方式*/
   if((fp=fopen("e:\\test.txt","rt"))==NULL)
   {
      printf("\nConnot open file!");
  getchar();
  exit(1);
   }
   ch=fgetc(fp);//从fp所指文件中取出第一个字符
   /*从文件中取出所有字符，一个字符一个字符的显示在中断上*/
   while(ch!=EOF)
   {
      putchar(ch);//输出一个字符
  ch=fgetc(fp);//取出一个字符
   }
   fclose(fp);
   printf("\n");
}
```

我们在 e:\\test.txt 下输入如下内容然后保存。



然后执行程序,结果如下



【注意】程序中的路径一定要写正确了

Fgetc 函数调用时，读取文件的方式必须是以读或者是读写的方式打开的。

（2）fputc()字符写入函数

【格式】fputc(ch,fp)

【含义】将字符 ch 写到文件指针 fp 所指的文件中。

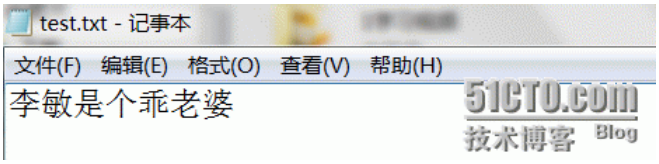
【返回值】如果写入成功，返回写入的内容 ch,如果输出失败，则返回一个 EOF 值。

【程序实例】

```
#include<stdio.h>
#include<stdlib.h>
main()
{
   FILE *fp;//定义文件指针
   char ch;
   /*判断文件打开是否成功*/
   /*打开方式为wt,写文件方式*/
   if((fp=fopen("e:\\test.txt","wt+"))==NULL)
   {
      printf("\nConnot open file!");
  getchar();
  exit(1);
   }
    /*输入的字符一个一个的写入到文件中*/
   printf("请输入一个字符串\n");
   ch=getchar();
   while(ch!='\n')
   {
      fputc(ch,fp);//将ch写入到文件中
  ch=getchar();
   }
   rewind(fp);//将文件的指针指向文件开头
   /*用fgetc来输出文件的内容*/
   printf("写入文件的内容是: \n");
   ch=fgetc(fp);
   while(ch!=EOF)
   {
      putchar(ch);
  ch=fgetc(fp);
   }
   printf("\n");
   fclose(fp);
}
```



好了，程序显示内容写入 test.txt 文件中并且输出了，我们找到 e:\\test.txt 下，打开 test.txt 看看是否有“李敏是个乖老婆”这句话。



可以看到的确是有了。

【注意事项】

在上述程序中我们可以注意到这么一个问题，test.txt 文件中本来就有了其它的内容（赵玉强的博客 http://zhaoyuqiang.blog.51cto.com），但是执行完上述程序后，文件中原来的内容不见了，取而代之的是新的内容（李敏是个乖老婆），如果想原来内容不删除的话，访问方式应该改为 a+(追加读写方式)打开。

防止人为误操作 MySQL 数据库技巧一例

作者：老男孩 oldboy 来自：http://oldboy.blog.51cto.com/2561410/1321061



IT/Linux 运维专家 老男孩
专注于 IT 运维领域 10 年以上，擅长大规模网站集群运维架构技术细节，长期兼职多家企业技术顾问。2007 年开办国内首个 Linux 运维实战培训私塾式精英教育培训机构。截止到 2012 年，累计受益学生达到千余人。
2011 年度 IT 博客大赛 10 强得主。
2012 年度 IT 博客大赛星光评委。

防止人为误操作 MySQL 数据库技巧一例
（本题来自老男孩培训内部学生问题，属于数据库安全技巧）

在若干年前，老男孩亲自遇到一个“命案”，老大登录数据库 update 一个记录，结果忘了加 where，于是悲剧发生了，这使得我对 MySQL 的增量恢复熟练度远超过其他的知识点，同时也很在意对内的数据库安全，并且每次讲课讲到此处，都会给学生讲这个领导的故事。

1、mysql 帮助说明

```
[oldboy_c64 ~]# mysql --help|grep dummy
 -U, --i-am-a-dummy Synonym for option --safe-updates, -U.
i-am-a-dummy      FALSE
```

在 mysql 命令加上选项-U 后，当发出没有 WHERE 或 LIMIT 关键字的 UPDATE 或 DELETE 时，mysql 程序就会拒绝执行

2、指定-U 登录测试

```
[oldboy_c64 ~]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -U
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.5.32-log MySQL Community Server (GPL)
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> delete from oldboy.student;
ERROR 1175 (HY000): You are using safe update mode and you tried to update a table without a
WHERE that uses a KEY column
mysql> quit
Bye
```

提示：不加条件无法删除，目的达到。

3、做成别名防止老大和 DBA 误操作

```
[oldboy_c64 ~]# alias mysql='mysql -U'
[oldboy_c64 ~]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 5.5.32-log MySQL Community Server (GPL)
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> delete from oldboy.student;
ERROR 1175 (HY000): You are using safe update mode and you tried to update a table without a
WHERE that uses a KEY column
mysql> delete from oldboy.student where Sno=5;
Query OK, 1 row affected (0.02 sec)
mysql> quit
Bye
[oldboy_c64 ~]# echo "alias mysql='mysql -U'" >>/etc/profile
[oldboy_c64 ~]# . /etc/profile
[oldboy_c64 ~]# tail -1 /etc/profile
alias mysql='mysql -U'
```

结论：

在 mysql 命令加上选项-U 后，当发出没有 WHERE 或 LIMIT 关键字的 UPDATE 或 DELETE 时，mysql 程序拒绝执行。

论“软件测试实施”

作者：hblxp321 来自：http://starpoint.blog.51cto.com/968349/1308715



软件测试专家 李晓鹏
河北师大软件学院测试教室主任、项目基地测试经理。擅长项目模块化流程设计、软件测试流程设计及优化、测试管理工具等技术；获得证书 MCITP、国家软件测评师、Oracle 数据库 DBA 认证（OCP）及华为高级网络工程师；出版图书《软件功能测试：基于 QuickTest Professional 应用》等。

测试实施阶段

当设计好用例、数据准备好后，就可以将设计好的成果应用于软件，如果在执行用例的过程中，发现软件执行的结果和测试用例中的预期结果不一致，那么，这就是软件缺陷即我们通常听到的软件 bug，我们需要把该缺陷报告给相关开发人员进行修改，如何报告呢，即需要记录缺陷，也就是我们通常所说的缺陷报告。那么缺陷报告包含哪些内容？

缺陷报告模板

编号：001		
软件名称：记事本	版本号：V1.0	
测试人员：**	日期：2013 年 8 月 30 日	指定处理人：
浏览器：IE 6.0	操作系统：Windows 2003	
严重程度：功能问题（高）		
优先级：P2		
缺陷概述：使用“记事本”仅保存“联通”二字后再打开该文件，出现乱码		
详细描述： 1 点击“开始”->“程序”->“附件”->“记事本”打开记事本软件； 2 仅输入“联通”二字后，点击“文件”->“保存”； 3 在打开的“另存为”对话框中保存文件后退出（文件名、保存位置任意）； 4 打开保存的文件，出现乱码，不是“联通”二字。		

51CTO.com 技术博客 Blog

缺陷报告通常包含如表所示的内容：

缺陷报告模板

此模板仅为众多缺陷报告模板中的一种，并不是缺陷报告的标准。读者应根据公司的实际情况来制定合理的缺陷报告模板。

以下，针对严重程度和优先级来进一步阐述。

缺陷严重程度：表示软件缺陷所造成的危害的恶劣程度 。缺陷严重程度的分类在不同的软件公司，分类有的也不同，在此给出一种分类，内容如下：

Fatal：致命的错误，造成系统或应用程序崩溃、死机、系统悬挂，或造成数据丢失、主要功能完全丧失等。

Critical：严重错误，主要指功能或特性没有实现，主

要功能部分丧失，次要功能完全丧失，或致命的错误声明。

Major：主要错误，这样的缺陷虽然不影响系统的使用，但没有很好的实现功能，没有达到预期效果。如提示信息不太准确，或用户界面差，操作时间长等。

Minor：一些小问题，对功能几乎没有影响，产品及属性仍可使用。

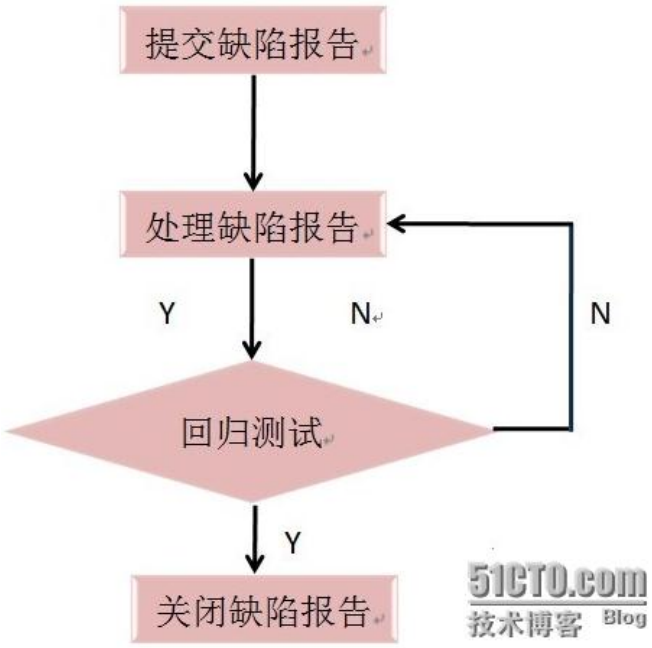
Suggestion：一些友好的建议。

缺陷优先级：表示修复缺陷的先后次序的指标。优先级的分类在不同的软件公司，分类也不尽相同，一般优先级的划分用 ABCD 或数字 1—4 表示，A 或 1 表示最高级别，D 或 4 表示最低级别。

- n 最高优先级：立即修复，停止进一步测试。
- n 次高优先级：在产品发布之前必须修复。
- n 中等优先级：如果时间允许应该修复。
- n 最低优先级：可能会修复，但是也能不修复。

缺陷报告流转过程

将缺陷报告提交给相应的开发人员，最终目的是为了能够进行缺陷的修复。那么软件缺陷究竟要经过怎样一个流转过程呢？以下，结合图所示的流程进行介绍。



缺陷流转过程

通常，软件缺陷的处理需经过如下过程：

1）软件测试人员提交缺陷报告；

2）测试经理审核后将缺陷报告分配给相关的开发人员修改；

3）缺陷被修改后由测试人员根据缺陷报告中的修改记录进行返测；

4）返测通过的缺陷报告由负责人关闭，返测未通过的缺陷报告直接返回开发人员重修修改，缺陷报告直到缺陷被修复以后才关闭。

以上仅是描述了一种缺陷报告的处理流程，在实际工中应根据实际情况进行处理流程的调整。

查看博客首页：
http://starpoint.blog.51cto.com/
查看相关视频：
http://edu.51cto.com/lecturer/user_id-968349.html
查看相关专题：
http://blog.51cto.com/zt/534

葵花宝典与学习之如何平稳的进入新技术领域

作者：IT 之梦 来自：http://itmydream.blog.51cto.com/961933/1294029

三个月了，三个月没有写过东西了，提起笔，不知道如何写这个开头，不管怎么样我还是动笔了，开始记录生活工作的点点滴滴，也算是好事的开始！

从 6 月份开始忙事情，忙了好久，终于可以先喘口气了！三个月时间，不长也不短，但从中得到的却很多，由于信息量比较大，这里我就分多篇来记录，这一篇我们分享面对一个我们未知的技术，我们该如何的去学习？只是个人这段时间摸索的经验，还请大家指正

对于未知的技术知识，由于我们存在一个未知的状态，我们往往不知道该如何的去攻坚他，如何牢固的去掌握他，如何的学以致用，那么我个人觉得，对于众多的新知识，我们需要以下导向来加以选择，不可盲目：

一、兴趣

无可厚非，兴趣绝对是我们学习任何知识的第一导师，我想我也没必要在这里大喷吐沫星子了，浪费大家的时间我担当不起的！但要记住，要符合所在企业的一些需求，不然要么你舍弃自己的兴趣，要么自己离职在进入一家和兴趣匹配的公司，但这比较奢望了！

二、工作需要

除开兴趣之外，工作需要也是我们学习的一大动力，因为我们要生存，我们就必须接受新知识的折腾，折腾完了就可以长久收益了，等待下一次的折腾，可能有人会觉得我现在过的好好的呀，没必要去尝试新方式了，这个社会是讲究效率的，你每次装系统还在用光盘/U 盘吗？软件升级你还要一个一个的去客户端升级吗？客户端有问题，你还需要跑过去调试吗？你还在为了年度资产统计去每个部门抄数据吗？你还在企业里充当救火员的角色吗？那么你凹凸曼了，因为技术也是有迭代的，你终将也会被淘汰，不是危言耸听，看到这里，你心里是否在嘀咕着应该如何改变现状了吗？如果是，那么我们该学习点什么新技术来提高效率呢？所以我们要先提出自己的需求，那么才会有动力去提出解决方案！个人建议，优先选择我们经常需要用到的技术进行优化，比如系统部署，软件安装，资产管理等，就像系统部署，重装系统等都是我们平常维护的时候经常遇到的，不可能你还拿个光盘/U 盘去 PE 启动然后拷贝用户数据，在进行安装吧！赶紧的改改吧，太 out 了，为了优化，我们可以选择部署 WDS/MDT/SCCM，别跟我说要收费，那不是借口，况且 WDS 和 MDT 已经足够中小型企业使用了，也是免费的，你值得从这一技术先研究做起（不管紧要的技术就先放一放吧）！因为他贴合实际，又有实际环境检验你的研究，当你发现部署了以后效率提高了很多，会更有信心去改变其他的现有技术实施方式！不仅在这里收益，到其他企业一样可以收益，永远不要落后微软太远，不然你就不得不抛弃微软！

以上只是提示我们，要合理的提出自己的需求，这样才会有针对性的解决方案，别想有万能的医者一药治愈！下面我就简单的探讨一下这段时间我研究 WDS/MDT 时候的一些见解

刚开始对 WDS/MDT 研究的时候，也是太盲目，有点找不着北的感觉，不知道从哪里找资料，不知道如何去辨证资料真伪，不知道。。。太多的不知道！但冷静下来好好的理下思路！
我们还是可以整理出一个途径的

收集资料

对于未知的技术，我们首先要得到一些资料，弄清楚是否满足我们的需求，方向不对，努力白费，所以，在选择资料方面。我们一定要仔细，除了官方的资料外，我们也可以选择一些实战型的资料，毕竟官方的资料比较死板，实战性的比如博客，论坛等其他一些

靠谱的人分享的，这里我推荐你使用 OneNote 这个软件，他的记录方式我很喜欢，具体的大家可以去 TechNet 上去看这个普及视频，可以截图看下（但这不是重点），通过一个方案的收集，我们
可以整理出一个技术点的实施，功能等我们想知道的内容



实际操作

根据我们上一步收集的资料，我们要跟着资料进行多次的实验，10 次 8 次你都别好意思说你做过，就针对 MDT 来说，我也做过很多次实验了，并且都是用真机操作，配置了 2 台台式机和一台笔记本，好处也是明显的，因为在部署的时候如果使用虚拟机，可以不考虑网卡驱动，但用真实机器，你就必须乖乖的导入驱动等，也算是一个全面的实验！没有机器环境，这个要自己来解决了，该花钱的别掖着！这里需要特别说明一点，拿到资料，别还没做几遍呢遇到问题就去群里，论坛发问，对于刚接触新知识遇到问题，做实验过不去很正常的，不要一遇到问题就去问（问了自己也不会明白为什么，而且也记不住），我还是建议你立马推到环境重新根据资料进行部署（这样可以逼迫自己多做几遍，熟记成功的操作，对比错误的操作），因为别人分享的博客都是经过多次试验一步一步写出来的，跟着资料作肯定会有结果出来，不可能问题一大堆的（当然，说的有点绝对，大家勿喷，我这里是想强调个人的因素，排除外在因素），首先应该找自身的原因，是否是自己哪一步没仔细看等等，说实话，在研究 MDT 的时候，我也遇到了很多次障碍，我没有去问楼主或者其他人，因为我知道相对来说这技术已经成熟了，不可能有那么多的 BUG，别人做的都没事，单单到我这里就是问题百出，出问题不可怕，那是我们还不熟悉如何操作！我会一遍一遍的去推到环境重新部署，最终还是一一告破，虽然不知道为什么，但结果我可以做出来，自信心还是增强了！

整理总结

经过无数次的操作，结果已经成功了很多次，那么你应该着手开始理解其中的原理，其中的部署技巧，以及需要细心的地方，如果可以最好自己坐出一套操作指导书，在整理的阶段，你可以大胆的去问，直到你心里没有死角为止，因为你已经学习过了，操作也会了，在看别人回复答案的时候也会很轻松（参考下这个帖子：http://bbs.51cto.com/thread-1072230-1.html 回答一个会出来三个不懂的问题，明显会感觉到一些新手对于一些自己不熟悉的技术发问，告诉他如何检查，如何排除，他给你来一个我不懂啊，没听过你回答的一些名词啊，你感觉如何？肯定会很崩溃！）何为学问？肯定是你学了以后不明白的在来问嘛，还没入门就开始问如何恢复一个账号，如何如何？就算我知道，我告诉你恢复的时候一定要用授权还原（肯定又要问什么是授权还原了！）这样不是做学问，是矫情！是对对方的不尊重，更是对自己的藐视！

总结：
其实我们生活中也有很多这样的例子：

小孩学话（本来是学习英语的模板，我们就先用到这里，到写英语那篇在详细写）

我们小的时候到了说话的年龄，我们懂什么？肯定啥都不懂对吧！但我们还是跟着大人们支支吾吾的说起来了，慢慢的象“爸爸”“妈妈”这样的字，我们都会说了，但我们知道这些字的含义吗？肯定不知道，只是会喊，在后面就学会了其他的字，比如自己的名字，在后来我们上学了，我们开始理解爸爸/妈妈/自己的名字等等他们的含义，都代表着啥，其实，这一路看起来跟我们上面总结的如出一辙，先是小孩找学习的对像（好比我们先收集资料），然后小孩跟着大人们学话，这个时候学话肯定是很不标准的，就跟我们拿着资料作试验一样，肯定会有很多的障碍，但不要害怕，一次学话学不好，我们会学第二

次，慢慢的从模糊的喊爸爸到清楚的喊出爸爸！在后面上学了，老师开始帮我们总结这些字的含义，我们也就开始记住这些，并作为现在我们人际交往的基础！你会发现老师讲完我们已经会的字的含义的时候你会很快记住，这就是优势，如果有十个美女，你见过其中一个人（但不知道她叫什么）这个时候有人给你介绍，那么你记得最清楚的肯定是你见过的人，这就是我们记忆的奥妙！技术也一样，你不会操作就想懂原理（就像考驾照，老师只给你讲原理，不给你操作，你能过考试？），就算懂我也只能说你是书本上的懂，实际都是狗 P，但如果你会操作，最后给你讲为什么这一步这么选择以及他们的工作原理（如果你开车早就会了，还没有学习书本只是，那么现在告诉你高速上要保持最少 100 米的距离，我想你很容易理解为什么要这么做），我想你很快就会贯通！

处对象

一个阳光明媚的上午，你走到步行街上，看到一个美女，那么你应该怎么办？

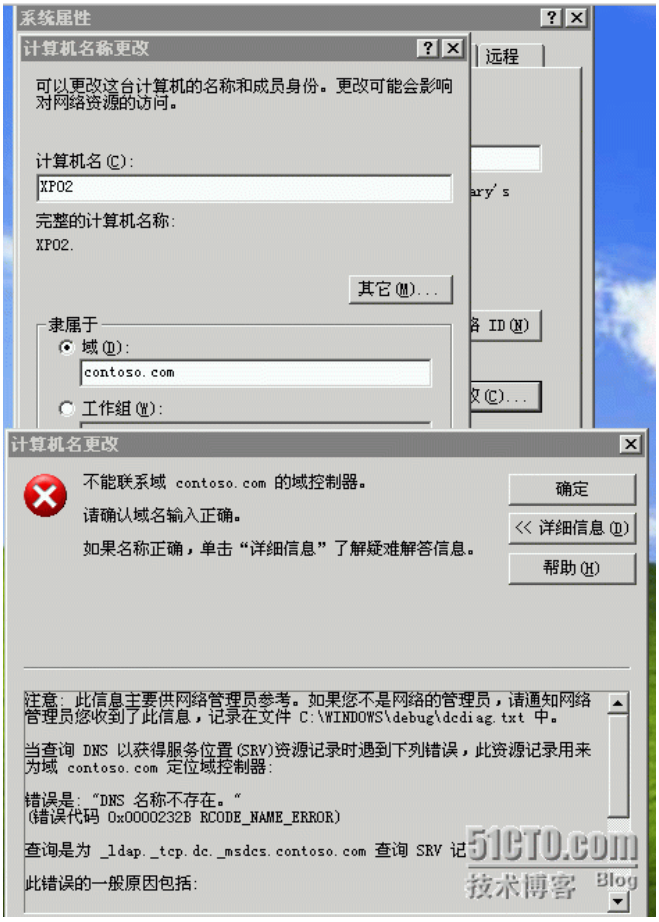
首先搭讪索取一些信息，然后人肉出来详细信息，如果已经结婚了，那么你也就放弃了，如果没有男朋友的，那么你会进一步的去接触，经过长久的接触，从未知到知心，你们也该准备结婚了，也该好好总结你们这段姻缘了！那么结合我们的总结，会不会感觉很像！

首先是收集美女的信息，筛选我们需要的，摒弃不需要的，然后好好的跟对方多次的接触，有的时候不止是多次，而是夜以继日的接触，经过一段时间的相处，才考虑结婚嘛，然后就是为这段姻缘做出一个总结哈！

其实说这些都是告诉我们，做事情肯定会有一个标准的流程，如果你根据流程来，事情会很顺利，但如果不根据流程来，那么事情可能就要绕弯来解决了！我们是最熟悉自己的，只有自己发现自己才会提高自己，单纯的依靠外在因素影响自己那时很难的，仔细一下，留心一点，你会发现，自己可以改变的还有很多！

学习也是有技巧的，本篇已经很长了我就简短一下：

我们如果按照正常的操作流程来操作的话，出错的几率很小，但这不是我们特别想要的，不出问题就是问题，我想大家应该明白，实验时候出现的问题比实际生产环境出现问题会好的多！我们在操作的时候，很多文档或者资料都会提示我们有些选项一定要这么操作，如果你恨乖，你会跟着操作，但如果出问题了你排查的时候可能需要很久，但如果这个时候你反向的操作一下，比如说，AD 环境，客户端的 DNS 需要指向 AD 的 dns 服务器，以此来解析，这个我想大家都懂，但这个时候如果你不指向 DNS，你不填或者随便填，那么肯定会报错如图，



这个时候对于工作原理的理解会有很大的帮助，这比单单的给你说 AD 是通过 DNS 来定位的更形象更深刻，你说对不对？所以，在标准流程做熟练之

后，可以尝试的进行反向的操作，然后把错误信息截图并作出正确解决方案进行总结成稿，相信你成长会很快！

附录

葵花宝典-新解

对于我们这一代人来说，金庸的武侠小说《笑傲江湖》那是火透了，而最令我们印象深刻的莫过于其中的一门绝学《葵花宝典》，练功时，首先要自宫净身，而东方不败做到了成就了她的霸业！对于葵花宝典，网络上大多都是很调侃的，好像是一种贬义的词性！其实对于《葵花宝典》的另一理解也是源于群里一个哥们对自身的一个描述：自己为了专心学习技术和英语，网络已经断了 2 个月了，他自己也总结了这 2 个月学到了很多！由他的故事我联想到了葵花宝典，我们不要局限的去理解葵花宝典的“自宫”，对比他和东方不败的情景，一个只是在精神上“自宫”而另一个是在精神上之外还有肉体上的“自宫”，我们不可想象肉体上的“自宫”，但我们可以想象精神上的自宫，用佛道说都是心无杂念，专心致志！

对于我们这一代人，网络横行，我们的精神支持大多都在网络上，我们的心思已经被怒放了，不可能会心无旁骛的去专心的做一件事情，QQ，微博，空间，维信等等，当我们去学习的时候，我们往往会被这些诱惑着，总是情不自禁的坠入他们的深渊，就像上面我提到的那哥们，自己断网 2 个月，我想对于现在的都市网络生活，与世隔绝 2 个月，他的这自宫换来的不是我等吊丝能够想象的到的，我想你对于成功的渴望对于这种自宫，你又有什么理由可以拒绝，又有什么理由去嘲笑，又有什么理由去贬低呢？

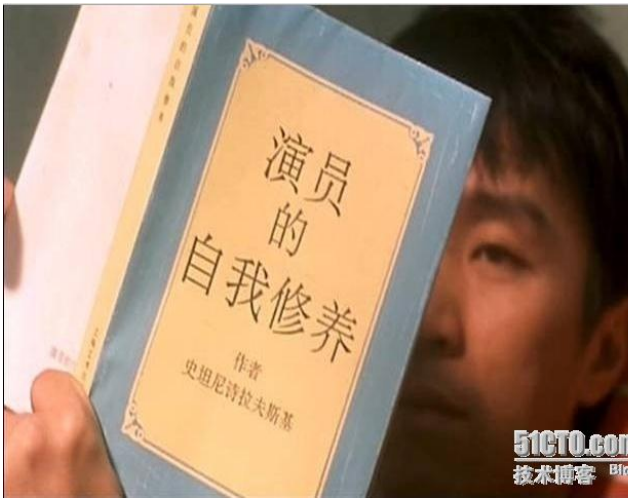
你的成功源于你勇于自宫！



查看他的首页：
http://itmydream.blog.51cto.com/
查看他的视频课程：
http://edu.51cto.com/lecturer/user_id-961933.html

那些逝去的专业软件研发精神

作者：大卫张 来自：http://davidzhang33.blog.51cto.com/3095817/1302952



我们最近的多次软件发布都出现了问题。或是因为某个代码低级错误发生回滚；或是因为 SQL 未仔细检查导致发布出现线上问题；或者是流程响应不及时导致客户业务不可用；甚至还有 Copy 代码导致资损。

面对这种情况，团队处理了问题，进行了回顾总结，还制订了一些措施。你看，我们及时的处理了问题，纠正了错误。我们还战胜了自己，敢于把问题暴露出来，直面问题。这些问题的发生都不是存心故意，属于不小心的偶然遗漏，不是主观原因；或者只是我们相信了别人的代码；或者是规范执行的力度不够；或者是管理层不重视，因此有所疏忽。对此，我们也进行了深刻的反思，我们还制定了后续措施，例如“加强 Review”“加强规范执行”等等。借此，我们可以发现一些问题，推行一些新的措施，加强规范执行。这一切似乎都很正常，甚至让人感到开心。

可为什么我却一点也开心不起来，羞愧、疲累、痛心，心中五味陈杂，辗转难眠？

不应该这样的！我们的发布应该很顺利，客户应该感受到新功能带来的好处并为之欢呼雀跃，为未来业务的发展充满憧憬展望。而不是像现在这样，或者是大为光火，重要的业务功能不能使用；或者是对我们将信将疑，不相信我们有能力履行承诺。我们没有给客户现在，更别奢望承诺将来。为此，我们可以找到很多理由原因，但我们无力解释，因为我们确实没有做好本该做好的发布。我为我们的表现感到羞愧，愧对我们对客户的承诺。

不应该这样的！我们应该顺利进入后续发布的研发，而不是花时间为上一个发布或者更早的发布搽屁股。有了这些时间，我们本该去关注那些超炫的新技术，并考虑如何应用到发布中，让其他技术团队羡慕嫉妒恨。有了这些时间，我们本该去考虑对客户有更大帮助的新功能，闪瞎客户的钛合金双眼。有了这些时间，我们本该去想想如何提升自己的技能，关注自己能力的成长，争取做一个有尊严的无证软件研发人员。而现在，我们的时间用在哪里？发布时通宵达旦，还要拜关公波多野结衣；发布后不断发现与解决问题，还要花不少时间沟通协调，推动数据订正发布回滚；更无聊的是，还要开什么缺陷分析会议，写什么故障总结，上面还非要写些我们明知道永远不会真正执行的 Action。我真心感到羞愧，我们的忙有很多都是自找的，我们浪费了自己的时间，成为了每天都在救火的救火队员。

不应该这样的！我们应该是擅长学习的、有尊严的软件研发人员。我们应该是最善于解决问题的人，我们本该能够找到方法，让之前出现的问题不会再度出现。我们应该是有尊严的软件研发人员，自己关注自己的饭碗，有能力把自己写代码的技能提升，真正对团队负责的发布、自己写的代码负责，保证不出低级问题，而不是像奴隶一样需要被别人用鞭子来规范、像码农一样被圈养。我们团队不少人都参加了对外面试，想想你们给别人的评语：“在公司这些年，只是完成了工作，没有太多学习思考，被荒废了，不满足我们的招聘需要”。再想想自己，曾几何时，我

们也已经失去了勇气，丧失了尊严。纯粹被工作推动，被动完成，不再敢为自己写的代码负责，开始为自己找到时间太紧等诸多理由；不再敢对不合理的要求抗争，都是被 PDPM 逼的，即使没被人逼，也要多做些东西，哪怕做错都可以。曾几何时，写出更好的代码，做更好的发布变成了其他人，尤其是主管、域专家、PM 的职责，我们开始放弃进公司时的聪明才智，开始随波逐流。只要管理者们不强调的东西，慢慢我们都不做了，因为这肯定不重要。我们开始忘记为自己负责，认为这都是别人逼我们做的，忘记了去写出漂亮的代码、去很好的理解业务、去把控质量有效交付，忘记了我们需要不断提升这些能力，忘记了因为这些能力我们才有被雇佣的价值。我觉得很疲累，不是因为发布出了问题，而是因为作为软件研发人员没能从发布问题中学习提升；我觉得很疲累，作为团队的负责人，是我没做好，没能唤醒团队同学的勇气和尊严；我也感受到了大家的疲累，大家虽然都想努力做更好的软件研发人员，却已经累了，不想再抗争，开始当一天和尚撞一天钟。

不应该这样的！我们应该共同为客户、产品负责，研发团队应该为发布负责。出现问题是我们大家的问题，没有人能置身事外。这个问题是不可能测试出来的，所以测试没错；都是某个同学不小心，规范执行不到位，所以都是他的错；反正领导也不重视，出错了也是不重视的原因；需求都没弄清楚，开发同学就做了，都是他的错；我是 PM、主管，代码我不熟悉，代码 Review 不关我的事。还有自测不充分、明知道问题也不提醒，反正有诸多错误，我们孜孜不倦的寻找他人的错误，致力于找出谁的错误更大，找出自己不在场的证据，动不动就想诉诸于暴力，仿佛惩罚了他人的错误，加强规范进一步奴隶化就能解决问题。这些有什么用？这就是我们对待同事、对待伙伴的态度？我们到底是不是一支团队？我们是想找人为错误负责，还是真的想向客户交付高质量的软件？我为此很是痛心，不是因为我们发布出了问题，而是因为不少同学置身事外，因为团队内的同学相互指责，更是团队外的同学麻木不仁。如果真的希望是伙伴，即使伙伴不完美，我们也应该相互提醒，共同去为客户交付更好的软件，小研发团队为发布负责，大研发团队为客户、产品负责。

作为软件研发人员，我们还能唤回职业的勇气与尊严吗，能够为自己而不是为别人工作吗？需要我要为团队做点什么，才能让团队摆脱现在的状态？作为团队的一员，你又能为团队做点什么？

深夜之中，卧床之上，脑海里又回想起那对我有重大影响的书《演员的自我修养》，耳边传来那振聋发聩的声音“其实，我是一个演员！”。不禁在想，如果我们每人心中都有一本《程序员的自我修养》，那该多好！

我的微软最有价值专家(Microsoft MVP)之路

作者：LearningHard 来自：http://learninghard.blog.51cto.com/6146675/1308386

前言

在这一年多的时间里，经常逛国内的一些技术社区和论坛，其中发现一个问题——现在很多朋友，大部分是学生和刚毕业参加工作的朋友，或工作 1-2 年的朋友都比较迷茫，不知道自己想要什么，想下决心努力学习但是又缺乏耐力坚持不了，所以经常会觉得非常迷茫。所以我决定写这篇文章来分享下本人成为微软最有价值专家（MVP）的一个经历，希望可以让处于迷茫的朋友有所感悟，并付诸行动地去寻找自己的目标并实现它。

问题剖析

对于坚持不了这个问题，主要原因还是缺乏明确的目标或者说定的目标太大太泛，因为有些朋友所定的目标都很大，例如：“我想成为技术大牛”，或“我想做架构师”、“我想获得赚月薪 2 万以上的工作”等这样的大而泛的目标，但是不可否定，这些目标都是非常好的，只是太大了，因为这些目标并不是一年 2 年就可以实现的，所以有些朋友坚持了一段时间后发现自己的技术和能力并没怎么提高，从而也就中途放弃了，这也就是坚持不了的原因，然而解决这个问题主要途径有两点：

一、心态，即调整自己那种快速求成，立竿见影的心态，因为很多收获都不可能立竿见影，即使可以那样的收获也是表面的。说到这里让我想到了 天龙八部里面的场景（因为最近有个台再重播黄日华版本的天龙八部，这几天也温习了下），其中鸠摩智强行练习少林寺 72 绝技和易筋经，表面上看上去已经练成，但是它已经深受内伤。所以首先要摆正心态。

二、就是重新定一个小而实际的目标，小而实际的目标即是那些可以比较容易达到和看到效果的目标。

因为一个人如果坚持一件事情很久仍然没有看到任何效果，即使有再好的心态也难免会心情低落没有心思继续坚持了。所以我们就要定一些小而实际的目标。例如：“今天背 10 个英文单词”、“这两天把值类型和引用类型搞懂”、“这三天理解下 C#中委托并写篇总结文章”等、等这些小而简单的目标实现了之后，又可以继续定一些稍大点的目标，即一个星期或两个星期可以实现的目标，例如： “实现一个断点下载程序”、“理解网络编程和实现一个简单 TCP 程序”等、如果这些需要一周或两周可以完成的目标达成之后，又可以定一些稍大的目标，即一个月或几个月可以完成的，如完成一本经典书籍的阅读并总结成博文，C# 推荐 CLR via C#，这样以此类推，等前面目标达成了之后再去定一些大点的目标，这样一个个小目标的完成也可以加强自己的信心，从而为完成大目标做好铺垫。说了这么多，可能有些朋友还是坚持不了，因为他觉得这太长了，然而如果这样的话，我觉得还是心态问题，所以说心态是首要的。问题剖析完了，下面分享下我的 MVP 之路（朋友也可以以 MVP 作为目标，这个属于一个稍大的目标了，因为属于年级别的了）。

我的学习之路

我是从 2012 年 6 月份加入博客园的，可以说从这时候才真真开始了我的.NET 学习之路，然而在大学的时候我已经接触了.NET 了，并且毕业设计也是用 Asp.net 完成的，但是在大学的时候，由于缺乏目标，不知道学这些东西都能做什么，所以每天也差不多都是玩游戏，考前临时复习一个月应付考试，记得当时宿舍考试前说的一句话就是：“我只求 60 分，不需要多”。四年时间过的很快，一转眼就大学毕业了，在校园招聘上找到了一家上海的企业也就过来上海实习了，当时做的是测试的工作，刚开始时做的是手动后面慢慢做自动化测试，然而在这工作的一年中，也就是 2011 年，我还是很迷茫，因为自己觉得很多懂不会并且很浮躁，静不下心来学习。然而在 2012 年 5 月份，我换了一家公司，刚进公司首先是

学习，在这段时间里面，我才真真意识到到自己对知识的渴望，其中当然也有看不下去的情况，但是没办法，因为这个是任务，所以也就坚持看了，后面发现自己对一些自己以前的知识点慢慢弄懂了之后，发现还是蛮有成就感的，所以后面也就养成看书和学习的习惯了。同时这段时间也经常看博客园中人写的文章，也学习到了很多。刚开始也只是看，并没有自己写文章来对自己学到的东西总结，主要是担心写的不好，不好意思发，但是后面想明白了——我写这文章也就是自己的一些学习笔记罢了，发出也是自己的一个总结，当然能帮助到其他人就更好了。后面也就渐渐养成了学习写博文总结的习惯了，并且博客园的氛围也非常好了，写完博文之后有很多热心的朋友评论支持，这也给我不少坚持下去的动力。在这一年中，我看过的书籍如下：CLR via C# 、深入理解 C#、你必 须知道的.NET、WCF 服务编程（这本书没看完，也没做写总结博文，现在差不多忘光了，所以写博文总结很重要）、深入浅出 WPF、精通.NET 互操作、Asp.NET 本质论等，期间也看过很多电子书，这里就不一一列出，注：红色标注的是重点需要看的书籍。

我的 MVP 之路



祝贺您！我们很高兴授予您 2013 年度 Microsoft® MVP 奖项！该奖项表彰您出色的技术社区领导者，他们积极与他人分享自己高超的实践专业知识和见解。我们感谢您过去一年在 Visual C#技术社区所做出的杰出贡献。

刚开始写博文也就是一个学习过程的总结，自己的目标也是定在月级别的，当时也没有定下 MVP 的目标，MVP 目标的定下主要受王清培兄弟的影响，之前通过博客园认识这位兄弟的，可以说他给我很大的鼓励和支持，然而在 2012 年 10 月的时候，王兄也如愿以偿地获得了 MVP 的荣誉，也真是这个时候，自己也定下了这个 MVP 的目标，经过这一年多的努力，我也终于把这个目标实现了。然而收到获得 MVP 这个消息的过程中出现了一个小插曲，本来 MVP 的通知消息应该在 10 月 1 号或者 2 号差不多可以收到美国微软 MVP 项目组的通知邮件的，但是不知道为什么？我的 QQ 邮箱却在 10 月 1 号没有收到 MVP 项目组邮件，当时就认为是没有申请成功，心情有点沮丧，但是国庆休息完来上班的时候却收到中国 MVP 项目组恭喜获得 MVP 荣誉称号的邮件，此时心情又开心又觉得纳闷（因为我开始认为自己没有获得的，这里又收到 MVP 项目组邮件），刚开始想会不会发错了？但是想想不至于啊，怎么可能会发错的，后面就请教了我认识的另外一位 MVP 和与中国 MVP 项目组的负责人确认了下，得到的回复是：“这次 MVP 名单上有你”，这时候才真真的开心起来了，然后继续问：“但是我没有收到没过微软那边发来的邮件，之前填写 QQ 邮箱有没有出现这种获得了没收到邮件的情况？” 得到的答复是：“以前也出现过，但是这次其他人用 QQ 邮箱也收到了 “。后面就叫她（中国 MVP 负责人）帮忙叫美国那边的同事再发一次，这样也就结束我们之间的对话了，不管怎样，最后结果是——我获得了这次 Visual C# MVP 的荣誉。同时这里特别感谢王清培兄弟、51CTO 的莉子和小米、博客园、51CTO 和 CSDN 这些社区以及哪些支持过我的朋友们，谢谢你们。

给迷茫朋友的一些建议

正如前面剖析部分介绍的那样，迷茫的朋友首先要调整心态，然后定一些小的目标，再去付诸行动去实现一个个小的目标，万事开头难，刚开始的时候都是艰难的，但是在坚持过程中遇到困难，这时候需要对自己说——刚开始都是这样的，慢慢来，继续学习。

培训写博文做总结的习惯，当然你也可以有其他方式的总结

可以多逛社区、论坛，这里包括国内和国外的，例如国内的社区有：博客园、CSDN、ITeye、51CTO 等；国外的有：Csharp Corner、MSDN 博客、codeProject。国内社区有：CSDN 论坛、51CTO 论坛和 MSDN 中文论坛等；国外的论坛有：stackoverflow、MSDN 等。

总结

希望这篇文章可以给大家一点帮助和感悟，总之学习没有捷径，只有多学习，多写代码，多总结这样反反复复的过程，相信坚持就是胜利。



扫描二维码，加 51CTO 博客为朋友

关注 51CTO 博客微信，打造你的个性！

http://blog.51cto.com

编后语

《51CTO 博客月刊》为 51CTO 博客整理出品
最终解释权归 51CTO.COM 所有

如果您有意投稿，请联系我们
如果您有反馈意见，请告诉我们

联系方式：
Email：blog@51cto.com
QQ: 1173854158

欢迎关注我们：
@51CTO 技术博客　　@中国 IT 博客大赛